UNIVERSITY*of*
TASMANIA

Practical Implementation of a Path Replanning System for an AUV Operating in Dynamic Environments

by

Hui Sheng Lim

B.Eng. (Marine and Offshore Engineering)

National Centre of Maritime Engineering and Hydrodynamics
Australian Maritime College

Submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

University of Tasmania

July 2021

Declaration of Originality and Statement of Authority of Access

This thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis, and to the best of my knowledge and belief it contains no material previously published or written by another person except where due acknowledgement is made in the text or the thesis, nor does the thesis contain any material that infringes copyright.

This thesis may be made available for loan and limited copying and communication in accordance with the Copyright Act 1968.

Signed:

Hui Sheng Lim

Date: 5 March 2021

Statement Regarding Published Work

The publishers of the papers comprising Chapters 2 to 6 hold the copyright for that content and access to the material should be sought from the respective journals/publishers. The remaining nonpublished content of the thesis may be made available for loan and limited copying and communication in accordance with the Statement of Access and the Copyright Act 1968.

These publications have been modified to fit into the structure of the thesis. The bibliographical details of the work are outlined below.

Paper 1. Lim, H. S., Fan, S., Chin, C. K. H., & Chai, S. (2018). Performance evaluation of particle swarm intelligence-based optimization techniques in a novel AUV path planner. IEEE OES Autonomous Underwater Vehicle Symposium, Porto, Portugal. http://doi.org/10.1109/AUV.2018.8729773.

Paper 2. Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., Bose, N., & Kim, E. (2019). Constrained path planning of autonomous underwater vehicle using selectively hybridized particle swarm optimization algorithms. *IFAC-PapersOnLine*, 52(21), 315–322. http://doi.org/10.1016/j.ifacol.2019.12.326.

Paper 3. Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., & Bose, N. (2020). Particle swarm optimization algorithms with selective differential evolution for AUV path planning. *International Journal of Robotics and Automation (IJRA)*, 9(2), 94–112. http://doi.org/10.11591/ijra.v9i2.pp94-112.

Paper 4. Lim, H. S., Chin, C. K. H., Chai, S., & Bose, N. (2020). Online AUV path replanning using quantum-behaved particle swarm optimization with selective differential evolution. *Computer Modeling in Engineering & Sciences*, 125(1), 33-50. http://doi.org/10.32604/cmes.2020.011648.

Paper 5. Lim, H. S., King, P., Chin, C. K. H., Chai, S., & Bose, N. (2021). Real-time implementation of an online path replanner for an AUV operating in a dynamic and unexplored environment (submitted and under the review of *Applied Ocean Research*).

Statement of Co-Authorship

The following people and institutions contributed to the publication of work undertaken as part of this thesis.

Candidate	Hui Sheng Lim	University of Tasmania
Author 1	Prof. Shuhong Chai	University of Tasmania
Author 2	Dr Christopher Chin	University of Tasmania
Author 3	Prof. Neil Bose	University of Tasmania, Memorial University of Newfoundland
Author 4	Dr Shuangshuang Fan	Sun Yat-sen University (Current), University of Tasmania (Previous)
Author 5	Peter King	University of Tasmania

Contribution of work by co-authors for each paper:

Paper 1: Located in Chapter 2.

Lim, H. S., Fan, S., Chin, C. K. H., & Chai, S. (2018). Performance evaluation of particle swarm intelligence-based optimization techniques in a novel AUV path planner. IEEE OES Autonomous Underwater Vehicle Symposium, Porto, Portugal.

Author contributions:

Conceived and designed the review: Candidate, Author 1, Author 2, Author 4

Developed the model: Candidate, Author 4

Performed the simulations: Candidate

Analysed the data: Candidate

Wrote the manuscript: Candidate

Paper 2: Located in Chapter 4.

Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., Bose, N., & Kim, E. (2019). Constrained path planning of autonomous underwater vehicle using selectively hybridized particle swarm optimization algorithms. *IFAC-PapersOnLine*, 52(21), 315–322.

Author contributions:

Conceived and designed the review: Candidate, Author 1, Author 2, Author 3, Author 4

Developed and performed the simulations: Candidate

Analysed the data: Candidate

Wrote the manuscript: Candidate

Presenting at the conference: Author 3

Paper 3: Located in Chapter 3.

Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., & Bose, N. (2020). Particle swarm optimization algorithms with selective differential evolution for AUV path planning. *International Journal of Robotics and Automation (IJRA)*, 9(2), 94–112.

Author contributions:

Conceived and designed the algorithms: Candidate, Author 1, Author 2, Author 3, Author 4

Developed and performed the simulations: Candidate

Analysed the data: Candidate

Wrote the manuscript: Candidate

Paper 4: Located in Chapter 5.

Lim, H. S., Chin, C. K. H., Chai, S., & Bose, N. (2020). Online AUV path replanning using quantum-behaved particle swarm optimization with selective differential evolution. *Computer Modeling in Engineering & Sciences*, 125(1), 33-50.

Author contributions:

Conceived and designed the algorithms: Candidate, Author 1, Author 2, Author 3, Author 4

Developed and performed the simulations: Candidate

Analysed the data: Candidate

Wrote the manuscript: Candidate

Paper 5: Located in Chapter 6.

Lim, H. S., King, P., Chin, C. K. H., Chai, S., & Bose, N. (2021). Real-time implementation of an online path replanner for an AUV operating in a dynamic and unexplored environment (submitted and under the review of *Applied Ocean Research*).

Author contributions:

Conceived and designed the algorithms: Candidate, Author 1, Author 2, Author 3, Author 4

Conceived, designed and performed the experiments: Candidate, Author 5

Analysed the data: Candidate

Wrote the manuscript: Candidate

We the undersigned agree with the above-stated proportion of work undertaken for each of the above published (or submitted) peer-reviewed manuscripts contributing to this thesis.

Signed:

Hui Sheng Lim	Shuhong Chai	Vikram Garaniya
Candidate	Primary Supervisor	Director
Australian Maritime College	Australian Maritime College	Australian Maritime College
University of Tasmania	University of Tasmania	University of Tasmania

Date: 5 March 2021

Acknowledgements

It still feels like yesterday when my primary supervisor, Professor Shuhong Chai, invited me to her office to discuss the possibility of me joining a PhD program. The journey of doing a PhD study was full of unexpected obstacles and unimaginable rewards. I have been very blessed to be able to meet and work with many amazing individuals who helped and motivated me towards the completion of this thesis.

First and foremost, I would like to express my sincere gratitude to my supervisory team for their constant support throughout the project. Although the team has changed several times during the course of this project, I am thankful to every past member of the team. Professor Shuhong Chai was not only an excellent academic mentor, but also a mother-like figure who always treats me like her own son and keeps me on the right track for a better future. I truly appreciate all the valuable life advice that she encouraged me with whenever I was dejected. Dr Christopher Chin has been both an inspirational mentor and a good friend to me; I cherish the time when we hung out in the town to de-stress after work. Although I did not spend much time in person with Professor Neil Bose, his support extended from Canada has never been lacking; our trip to the conference in Portugal was one of the best memories I had from this project. I am also grateful to Dr Shuangshuang Fan for her support before she left the university.

I would like to thank the Australian Maritime Systems Laboratory for supplying the data and technical information required for this study. In particular, a very special thanks goes to Peter King, who provided me with extensive guidance and assistance for the experiment work, as well as his insightful advice for improving the algorithms.

I am thankful to my PhD cohort for being great companions to get through ups and downs when we struggled with our PhD projects. Listening to each other's complaints about the problems we faced in our projects was one of the best ways to relieve stress (apart from playing video games together). I am also grateful to the staff from Dynasty Chinese Restaurant and fellow church members of Praise Methodist Church. We have spent so much time having fun together during breaks from study, so much so that I could not name the most memorable event here.

There are no words that can adequately express my gratitude to my beloved parents for their consistent and unconditional love. My parents place their top priority on our welfare and are always supportive of whatever career path that I choose. My gratitude also goes to my brother and sister, who take good care of dad and mum while I am away. I would also like to thank my lovely partner, Yee Von Teo, who has showered me with endless support and love throughout the years. She has always been the person I could turn to during every desperate moment. She literally contributed to this project by proofreading my publications and this thesis.

Last but not least, I would like to express my utmost gratitude and respect to Mr Geok Chwee Ng (Uncle Dua Bah), who motivated my parents to let me study abroad; may his kind soul rest in peace. Without him, it would not be possible for me to pursue my studies in Australia. I trust that I will always remember to pass on his legacy of kindness and compassion to every person I meet in my life.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship.

Abstract

Autonomous Underwater Vehicles (AUVs) are increasingly used for a wide variety of missions with extended durations. Due to the limited onboard battery capacities of typical AUVs, the energy shortage of AUVs is a relevant issue that requires further research to solve. Efficient motion and path planning is one of the key factors for completing long-duration missions. Traditional pre-generative planners are inadequate to adapt to unknown and dynamically varying ocean environments. This is particularly evident for long-duration missions, in which a vehicle may unexpectedly encounter adverse ocean currents and suffer energy shortages. As the technological development in recent years continues to improve the onboard computational power of AUVs, it is timely to explore the potential of a sophisticated path planner for improving the operability, efficiency and endurability of AUVs. This thesis focuses on the development of a practical online path planner to improve the performance of an AUV operating in dynamic environments.

The particle swarm optimization (PSO) algorithm and its variants are suitable for the application of online path planning in dynamic environments because they can maintain a large pool of solutions that can be used for replanning a vehicle path at any time throughout the mission. Nonetheless, there are concerns about the practicability of PSO-based path planners such as the convergence of particles at local minima due to limited time for online planning, as well as their computational loads when implemented in an actual vehicle. Therefore, a preliminary review was first conducted to identify the strengths and weaknesses of existing PSO-based algorithms for solving the AUV path planning problem. A pre-generative AUV path planner was developed to solve the offline path planning problem of an AUV operating in a turbulent and cluttered ocean environment. The path planner successfully generated safe and feasible time-optimal paths that can exploit ocean currents to improve the AUV's performance.

Based on the preliminary review, a new approach was proposed to improve the performance of a PSO-based path planner by using selective hybridization of differential evolution (DE). The novel algorithms can conduct the DE operation selectively on particles to enhance the particles' searching ability and resistance to local minima without inflating the computational cost. According to the results of Monte Carlo simulations and Kruskal-Wallis tests, the proposed algorithms outperformed other algorithms in the tested scenarios.

To improve the path planner's search efficiency, which is the most critical attribute for an online path planner, constrained optimization was applied to formulate the path planner. The search domain was modelled using the polar coordinate system and a combination of hard and soft constraints. This ensured the compliance of paths with the vehicular constraints and facilitated the placement of path nodes to improve search efficiency. Different types of constraints were analysed to identify the optimal constraint setting that produced the highest search efficiency.

Using a novel PSO-based algorithm, an online AUV path planner that employed a path replanning scheme was proposed to address the path planning problem of an AUV operating in a dynamic and unexplored ocean environment. The proposed path replanner can continuously refine a safe and feasible time-optimal path for an unknown environment based on the feedback from its onboard sensors. The performance and robustness of the path replanner were verified in the Monte Carlo simulations that used the REMUS 100 AUV model. Next, the path replanner was implemented in an AUV system by using an open-source system architecture, MOOS-IvP. The implemented path replanner was verified in a hardware-in-the-loop (HIL) test of an Explorer AUV. The path replanner seamlessly worked in conjunction with the hardware on the test platform in real time to continuously generate a time-optimal path.

Based on the experiments that involved different sensor configurations and test scenarios, the path replanner demonstrated its scalability for missions that required different setups of onboard sensors and for AUVs of different sizes. It also showed its versatility to accept different current profile data for the generation of time-optimal paths. The resultant path replanner developed in this thesis is practical and promotes ease of applications in field operations of AUVs. It contributes to enabling an AUV to achieve a higher level of autonomy and hence improve its competence in missions with longer durations.

Table of Contents

List of Figures

List of Tables

Abbreviation

2D	Two-dimensional
3D	Three-dimensional
ACO	Ant colony optimization
ADCP	Acoustic Doppler current profiler
AMC	Australian Maritime College
ANOVA	Analysis of variance
APF	Artificial potential field
APSO	Adaptive particle swarm optimization
AUV	Autonomous underwater vehicle
DE	Differential evolution
DEPSO	Differential evolution particle swarm optimization
DEQPSO	Differential evolution quantum-behaved particle swarm optimization
DOF	Degrees of freedom
FA	Firefly algorithm
FLS	Forward-looking sonar
FM	Fast marching algorithm
GA	Genetic algorithm
GNC	Guidance, navigation and control
H-ADCP	Horizontal acoustic Doppler current profiler
HIL	Hardware-in-the-loop
IvP	Interval programming
LOS	Line-of-sight
MOOS	Mission-oriented operating suite
PSO	Particle swarm optimization
QPSO	Quantum-behaved particle swarm optimization
RRT	Rapidly-exploring random tree
SDEAPSO	Selective differential evolution adaptive particle swarm optimization
SDEPSO	Selective differential evolution particle swarm optimization
SDEQPSO	Selective differential evolution quantum-behaved particle swarm optimization
UAV	Unmanned aerial vehicle
UTAS	University of Tasmania

Nomenclature

C_1, C_2	Acceleration coefficients of PSO
D	Number of dimensions of a particle in PSO
d_{safe}	Safety distance for obstacle avoidance
d_{buff}	Buffer distance for obstacle avoidance
F	Objective function
gbest	Global best state of a particle swarm in PSO
i	Particle index in PSO
L	Characteristic length of the potential well in QPSO
m	Number of waypoints that form a path
mbest	Mean best position of a particle swarm in QPSO
N	Population size of a particle swarm in PSO
N_s	Number of particles selected from a particle swarm
n	Number of path nodes that control the shape of a path
O	Obstacle in a problem space
p	Waypoint of a path
pbest	Personal best states of particles in PSO
S	Selection factor for the hybridization of DE
t	Iteration number
$t_{\max}$	Maximum number of iterations
T	Trial vector produced by crossover in DE
U	Donor vector produced by mutation in DE
V	Velocity vectors of particles in PSO
V_a	Water-referenced velocity of a vehicle
V_c	Velocity of ocean currents
V_g	Ground-referenced velocity of a vehicle
w	Inertia weight of PSO
X	Position vectors of particles in PSO
β	Contraction-expansion (CE) coefficient of QPSO
δ	Potential well of QPSO
ϕ, θ, ψ	Roll, pitch and yaw angles of a vehicle
r, Φ, Θ	Radial, azimuthal and polar coordinates

This page is intentionally left blank.

Chapter 1.

Introduction

1.1 Background

Autonomous underwater vehicles (AUVs) are unmanned robots that can be pre-programmed to conduct underwater missions without relying on direct controls from a human operator. AUVs have become a progressively more important tool for performing various operations, ranging from seabed surveys, coastal mapping, and environmental monitoring for scientific research purposes, to mine countermeasures and anti-submarine warfare for military applications. As the market demands AUVs to expand their range of applications, the vehicles are required to execute increasingly challenging missions in more dynamic and constrained environments over extended durations. The energy requirements of such missions have outpaced the technology development of AUVs' onboard battery capacities (Edwards et al., 2017). As typical AUVs have limited onboard resources, efficient motion and path planning becomes one of the key factors for completing missions that challenge the vehicle autonomy and endurance.

Path planning is the process of generating paths to be followed by a robotic platform in order to conduct its mission. In the context of AUVs and other underwater vehicles, the term "path planning" is used more commonly than "motion planning", which often refers to the procedure of determining suitable actions for ground or aerial robots to carry out their operations. An AUV path planner must satisfy several objectives and criteria in accordance with the properties of the vehicle and its operational space. The minimum requirement of a path planner is to generate a path that enables an AUV to traverse towards its target while maintaining a safe distance from obstacles and respecting its vehicular constraints. In addition to fulfilling the minimum requirement, a path planner can improve the performance and endurance of an AUV by exploiting ocean currents.

Spatiotemporal currents, which are present in any AUV mission, can have a profound impact on the vehicle's performance. Strong ocean currents and eddies may oppose the AUV's motions and even push the vehicle off its planned path, leading to an increase in its energy consumption and thus a reduction in its endurance. A path planner that takes into consideration the effect of ocean currents can generate a time-optimal path, which increases the operational efficiency of an AUV by guiding the vehicle towards its target within minimum time. By adapting its planning to ocean currents, a time-optimal path planner can enable an AUV to surf the favourable currents that assist the vehicle's motion, while avoiding the adverse currents that are opposing it. A time-optimal path is particularly important when the AUV is required to traverse between multiple regions of interest across large bodies of water and over an extended duration.

Due to the spatial and temporal variabilities of dynamic ocean environments, in which an AUV may encounter moving obstacles and time-varying ocean currents, a planned path may become less optimal or physically infeasible over time, especially for missions with extended duration. In order to ensure a safe and optimal path for an AUV operating in dynamically varying environments, path planning needs to be carried out online and continuously throughout the AUV's mission. Path replanning is a technique used to improve the computational efficiency of online path planning by reusing previously planned paths for the planning of a new path. This thesis focuses on the development and implementation of path planning and replanning techniques that generate time-optimal paths to exploit ocean currents in a dynamic, cluttered, and unknown environment.

1.1.1 Overview of Path Planning Techniques

A path planning system (or path planner) of an AUV usually works together with the vehicle's guidance, navigation, and control (GNC) systems to provide autonomy to the vehicle. The framework of an AUV's GNC systems incorporating a path planner can be described in Figure 1.1. After receiving target information from the mission planner, the path planner generates a feasible path based on the vehicular and environmental conditions, which are measured (or estimated) by the navigation system. The generated path can be fed to a line-of-sight (LOS) guidance scheme and a reference model to calculate the reference (desired) vehicle states for path following and tracking. The reference states are passed to the control system, which computes the control forces required for actuating the vehicle to follow the path.

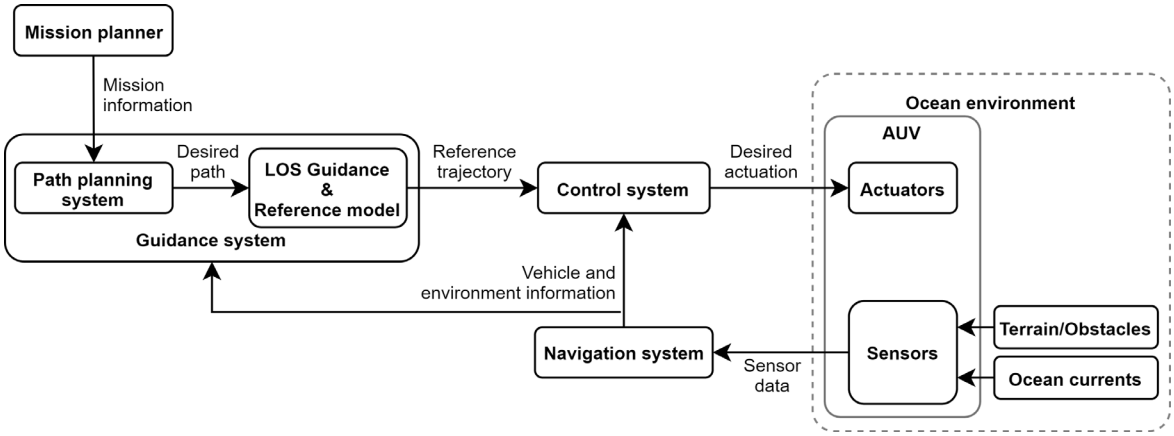


Figure 1.1: Path planning and GNC systems of an AUV.

Robotic path planning is an active area of research that has been extensively studied for many years, especially for ground vehicles. Path planning problems that involve lower dimensions and fewer constraints can simply be viewed as finding the shortest path to get a vehicle to its destination. However, the complexity of path planning problems can vary greatly due to the unique characteristics of different robotic platforms. Some robots, such as an unmanned aerial vehicle (UAV) or an AUV, are required to operate in higher-dimensional space with more degrees of freedom (DOF) and constraints. Path planning for an AUV should fulfil additional criteria such as the AUV's vehicular constraints, limited onboard computational power, and external forces in ocean environments. Moreover, robots in a real-world system are subjected to uncertainties that arise from their operational environments, GNC systems and actuators. In spite of extensive research conducted over the years, path planning problems with complex objectives and constraints remain computationally intractable. Generally, path planning can be treated as an optimization problem, which can be divided into two major components:

- Generation of a path constituted by a sequence of path nodes.
- Formation of a path by connecting the path nodes with a geometry.

1.1.2 Path Generation

To date, numerous AUV path planning techniques have been studied and applied to optimize the generation of path nodes. Earlier studies mostly focused on pre-generative offline planning (also known as global path planning), which assumed an *a priori* known environment and omitted the state information feedback from the navigation system. The offline planning approach can be applied for predictable environments, such as a previously

explored environment with known terrain and weather maps and subjected to little or no temporal variabilities. To generate a time-optimal path based on the offline planning approach, a predictive ocean model can be used.

Recent comparison studies from Zeng et al. (2016), Youakim and Ridao (2018), and Panda et al. (2020) classified and discussed various path planning techniques, such as potential field methods, graphical search-based methods, Markov decision process (MDP), finite-difference methods, sampling-based algorithms, and optimization-based methods. The potential field methods, also known as artificial potential fields (APF), are fast and efficient for high-dimensional problems but it is highly susceptible to local minima. Kruger et al. (2007) and Cheng et al. (2015) applied the APF method to generate a time-optimal path for an AUV based on a predictive ocean model. Graphical search-based planners, such as Dijkstra's algorithm, A* (Garau et al., 2009; Koay & Chitre, 2013) and Fast Marching* (FM*) (Petres et al., 2007), use low complexity algorithms with applications limited to lower-dimensional and less complex problems. Li et al. (2020) employed a Dijkstra-based approach to present a two-stage path planner that generates time-optimal paths based on the state constraints, reachability, and risk of collision. In order to improve the robustness of an offline A* path planner, Huynh et al. (2015) considered uncertainties in their ocean model to generate time-optimal paths.

Pereira et al. (2013) and Hollinger et al. (2016) incorporated uncertainties of ocean models to propose risk-aware MDP-based planners that produce robust pre-generative time-optimal paths. Time-optimal paths can also be generated by solving partial differential equations, in particular the Hamilton-Jacobi equation, with a finite-difference method. For instance, a fast sweeping method was employed by Takei and Tsai (2013) and Parkinson et al. (2020) to achieve time-optimal motion planning of a nonholonomic car. Lolla et al. (2012) and Lolla et al. (2014) used a level-set method and predictive ocean models to generate time-optimal paths for an AUV.

Rapidly-exploring random trees (RRT) (Cui et al., 2016; Rao & Williams, 2009) is a widely applied sampling-based method that is effective for high-dimensional and highly time-constraint scenarios, but its generated paths are usually suboptimal and require further refinement. More details on sampling-based path planners can be found in the study of Elbanhawi and Simic (2014).

Optimization-based methods can offer the advantage of optimizing multiple control objectives by using a combination of objective functions. The Covariant Hamilton Optimization Motion Planning (CHOMP) is an optimization algorithm that uses covariant gradient descent to achieve continuous path optimization. Another algorithm, the Stochastic Trajectory Optimization Motion Planning (STOMP), generates optimized paths based on a search from randomized solutions. It can offer higher resistance to local minima compared to CHOMP. Metaheuristic optimization algorithms, such as genetic algorithm (GA) (Alvarez et al., 2004), simulated annealing (Witt & Dunbabin, 2008), and particle swarm optimization (PSO) (Taleshian & Minagar, 2015; Wang et al., 2016), show excellent performance in terms of solution optimality for complex multimodal path planning problems and have high resistance to local minima, although these algorithms may converge to local minima within finite time. Fu et al. (2009) and Zeng et al. (2016) highlighted the efficiency and robustness of the quantum-behaved PSO (QPSO) algorithm for solving high-dimensional path planning problems.

1.1.3 Path Geometry

A vehicle path can be formed by connecting path nodes based on a chosen geometry, which can be a straight line, an arc, a spline, or a combination of the aforementioned (*i.e.*, a composite path). The formation of a path needs to take into account the dynamics and constraints of the vehicle. Forming a path using only straight lines produces an unsmooth path with a discontinuous first derivative at the location of every path node, resulting in an intermittent velocity function. A fully actuated robotic vehicle may be able to follow a path with some discontinuities easily. However, most aerial and marine vehicles, particularly an underactuated AUV, strictly require a smooth path to produce a satisfactory path-following performance. It is important for a vehicle to follow its path closely and pass through every waypoint because missing certain waypoints may cause the vehicle to collide with an obstacle, especially in a dynamic environment.

The continuity of velocity function along a path can be achieved by connecting the path nodes with a Dubins path, which gives the shortest path in a plane while fulfilling prescribed initial and terminal headings and a curvature constraint (Dubins, 1957). A Dubins path is a composite path that is formed by concatenating straight lines and circular arcs of maximum curvature. The traditional Dubins path considered only the forward motion of car-like robots in 2D but it was further studied by Reeds and Shepp (1990) to address the reverse motion,

and by Ambrosino et al. (2009) to generate 3D Dubins paths for a UAV. There are circumstances in which forming an optimal path from a Dubins path is not guaranteed to be feasible, such as when the external forces (wind or current) acting on a robot are considered (Bakolas & Tsiotras, 2013; Techy & Woolsey, 2009). Moreover, a Dubins path only has G^1 continuity and a discontinuous curvature function that jumps between zero and non-zero values due to the concatenation of straight lines and circular arcs. This increases the difficulty for a robot to follow the composite path due to the sudden change in the required centripetal acceleration for manoeuvring between the straight and circular segments.

A smoother path of continuous curvature (G^2 continuity) can be formed by concatenating segments that have the same curvature at their joining points. This can be achieved by joining straight lines and circular arcs with other geometries, such as a clothoid (also known as Euler spiral or Cornu spiral) (Fraichard & Scheuer, 2004). A clothoid path can have continuous curvature because the curvature of a clothoid starts at zero at its boundaries and varies linearly with its length based on Fresnel integrals. Clothoids have been applied in 2D path planning of AUVs (Barisic et al., 2010) and UAVs (Dai & Cochran, 2009; Shanmugavel et al., 2010). The application of clothoid-based paths is limited because solving the non-closed-form expressions of the Fresnel integrals can cause excessive computational loads for an autonomous vehicle (Dahl, 2013), especially in scenarios where online path planning is required.

An alternative geometry that can be used for generating a curvature-continuous composite path is a Fermat's spiral (also known as parabolic spiral), which also has a variable curvature that can start at zero at its boundaries. Fermat's spirals offer the advantage of lower computational cost because their solutions can be determined explicitly from simple parametric equations. The application of Fermat's spirals in path planning was first studied by Dahl (2013) and Candeloro et al. (2013). The parametrization of Fermat's spirals was further modified by Lekkas et al. (2013) to improve the trackability of generated paths. Subsequently, Candeloro et al. (2017) applied Fermat's spirals in a path replanner for dynamic path planning.

The abovementioned methods, namely Dubins, clothoid and Fermat's spiral, generate composite paths that consist of multiple concatenated segments. An alternative method to construct a feasible vehicle path is by using splines or other piecewise polynomial curves. In this case, path nodes serve as the control points for the curve functions. Although there

are a vast variety of piecewise polynomials that have been extensively studied and applied in computer science, few are suitable for path planning of underactuated robotic vehicles. Bezier curves (Elhoseny et al., 2018; Hassani & Lande, 2018; Zhang et al., 2016) and their generalizations, B-splines (also known as basis splines) (Chen et al., 2013; Wang et al., 2019; Zeng et al., 2012), are commonly used for path planning because they offer C^2 continuity and local control over the path, which means relocating one path node only affects the adjacent path segments. However, paths based on Bezier curves and B-splines do not pass through all their control points (Jolly et al., 2009); this restricts their applications in some path planning scenarios (Lekkas & Fossen, 2014). On the other hand, although natural splines have continuous curvature and pass through all their control points, they are not suitable for path planning because they lack local control and produce unnecessary wiggling between path nodes, leading to extra control actions required to follow the paths.

Another polynomial suitable for constructing a vehicle path is Pythagorean-hodograph (PH) curves, which are polynomial curves with hodographs that meet a Pythagorean condition. Farouki (2008) proved that a quintic PH curve was required for the path planning application to produce a closed-form path solution with continuous curvature and sufficient flexibility. PH paths in 2D and 3D scenarios were also studied for various autonomous robots (Choe et al., 2016; Farouki et al., 2018; Shanmugavel et al., 2007; Tsourdos et al., 2010). Cubic Hermite splines have also been employed for path planning (Bakdi et al., 2017; Song et al., 2015) to generate practical vehicle paths that offer high tractability without unnecessary wiggling segments. Paths based on the cubic splines pass through all the path nodes and allow local control. Despite not having a continuous second derivative (C^2 continuity), cubic Hermite spline paths can be applied for marine vehicles with a suitable path following controller (Lekkas & Fossen, 2014).

Due to the peculiarities of polynomial curves, the choice of curves for path formation depends on the types of vehicles and mission requirements. For underactuated vehicles such as typical torpedo-shaped AUVs, it is critical to ensure a planned path has a minimum of G^2 and C^1 continuity so that the path can be followed closely and smoothly. This is particularly important for an AUV subjected to spatiotemporal ocean currents, which increase the difficulty of path following and the likelihood of deviations from the planned path.

1.1.4 Online Path Planning

Online path planning pertains to the process where a path is continuously planned in real time for a vehicle subjected to dynamic variabilities. Thus, it is also referred to as real-time or dynamic path planning. Online path planning enables a vehicle to adapt to unexpected changes in a dynamic and/or unknown environment, which is a common operational scenario for real-world planning problems. The main challenge of online path planning is finding an optimal path within a limited time allowed for planning. A practical online path planner should achieve a balance between its computational requirement and the quality of generated paths.

An intuitive approach to online planning is known as local path planning, in which local adjustments are made for a vehicle while following a previously planned path. Local path planning mainly involves avoiding collisions with an obstacle and guiding the vehicle back to the planned path after passing the obstacle. Previous studies (Casalino et al., 2009; Larson et al., 2006; Yao et al., 2020) combined path following and obstacles avoidance control to handle dynamic environment effectively but at the cost of path optimality. Based on a similar approach, Sun et al. (2018) were able to improve path quality by combining fuzzy control and the QPSO algorithm. Benjamin et al. (2019) proposed a dynamic obstacle manager that uses multi-objective optimization of interval programming to ensure a collision-free trajectory while following a pre-planned path.

Some studies adopted a reactive path planning approach, in which a new path is generated reactively to adapt to the varying environment while previously planned paths are discarded. Existing studies proposed to combine the reactive approach with various algorithms such as Voronoi diagram (Candeloro et al., 2017), APF (Haddadin et al., 2010; Petres et al., 2011), A* (Duchoň et al., 2014; Naeem et al., 2012; Singh et al., 2018), Field D* (Carsten et al., 2006), and RRT (Redding et al., 2007; Vasile & Belta, 2014). Belkhouche (2009) and Belkhouche and Bendjilali (2012) also proposed reactive path planners that use simple collision cones and kinematic-based navigation laws to generate suboptimal but safe and robust paths.

Reactive planning approach was also applied with neural network and reinforcement learning to generate safe AUV paths in dynamic environments (Cui et al., 2017; Duguleana & Mogan, 2016; Lin et al., 2019). Cheng and Zhang (2018) proposed a deep reinforcement

learning algorithm that combined multiple reward functions to achieve effective obstacle avoidance in unknown environments, but it did not demonstrate significant advantages over non-learning algorithms. Although these learning algorithms showed their effectiveness in numerical simulations, their implementations in AUVs are challenging because model training for an AUV in the real world is expensive and time-consuming. Training a model in simulations is possible but the model will become biased and very specific to the simulated environments. None of the abovementioned path planners was verified experimentally on an actual AUV platform.

In contrast to reactive path planning, an approach known as path replanning scheme generates a new path based on the previous solution(s). In some literature, the term "replanning" is interpreted as redoing the planning and can be used interchangeably with reactive path planning. In this thesis, path replanning refers specifically to the technique of reusing previous solution(s) for the generation of a new path. Usually in oceans, the environmental conditions change gradually rather than transform completely and abruptly. The planning conditions for a new path may bear some resemblance to the conditions in the previous planning cycle. Hence, it is deemed more computationally efficient to generate a new path by modifying the previously planned path(s) because it is probable that the new path nodes can be placed in proximity to the previous solution(s). The increase in computational efficiency by making use of the previous solution(s) can be significant especially when the search space is vast and highly dynamic.

Previous studies proposed path replanning based on a single previous solution. For example, an "anytime" approach was developed to generate a feasible but suboptimal path that can be modified and refined continuously throughout the mission. This approach was combined with A* (Likhachev et al., 2004), Field D* (Ferguson & Stentz, 2006b) and RRT (Ferguson & Stentz, 2006a). Park et al. (2013) adopted a similar approach and developed a path replanner that uses parallel computing on multi-core processors to improve its computational performance. Other algorithms were also used to replan a new path from a previous solution, such as D* lite (Sun & Zhu, 2016), STOMP (Galceran et al., 2015), and RRT (Hernández et al., 2019; Ma et al., 2018).

Path replanning can also be conducted by reusing more than one previous solution. Bruce and Veloso (2002) developed an RRT-based path replanner that stores the cache of all previous waypoints for path replanning. To enable path replanning from multiple solutions,

existing studies also proposed the use of homotopic sets of paths with APF (Brock & Khatib, 2000) and RRT (Hernández et al., 2011).

Path replanning based on a pool of previous solutions can also be achieved by using a population-based optimization algorithm, which can maintain all the previous solutions at any time throughout a mission. For example, MahmoudZadeh et al. (2018) successfully applied PSO, differential evolution (DE), firefly algorithm (FA) and biogeography-based optimization (BBO) for path replanning. The application of PSO in path replanning was explored by several recent studies (Biswas et al., 2016; Lv et al., 2019; Zhou et al., 2018). Zeng et al. (2015) also proposed a QPSO-based path replanner to replan the path of an AUV in spatiotemporal environments at a predefined fixed interval. This path replanner has a high requirement for onboard sensor configurations because it requires either the global environmental information or all the information surrounding the AUV up to a specific radius to be known.

Although the PSO algorithm and its variants offer an effective and computationally efficient solution for online path planning, there are concerns about the practicability of PSO-based path planners such as the convergence of PSO at local minima due to limited time for online planning, as well as the computational loads when implemented in an actual vehicle. The majority of previous studies for PSO-based planners are only based on pure numerical simulations. Hence, it is critically important to study the performance of PSO-based path planners when implemented in real-time vehicle systems.

1.2 Research Aim and Objectives

The literature review in the previous section reveals that the path planning of an AUV in an unknown and dynamic environment is a computationally intractable problem that remains an active area of research. As AUVs are increasingly used for missions with extended durations, the shortage of onboard energy is a relevant issue that requires further research to solve. Pre-generative planners are inadequate to adapt to unknown and dynamically changing ocean environments. This is most notably evident for long-duration missions, in which the vehicle may unexpectedly encounter adverse ocean currents and suffer energy shortages.

Technological advancement in recent years continues to improve computers' speed and form factor. This enables an AUV to be equipped with higher onboard computational power

and allows it to achieve a potentially higher level of autonomy. Thus, it is timely to explore the potential of a sophisticated path planner for improving the operability, efficiency and endurability of an AUV.

This thesis aims to develop a practical online path planner to improve the performance of an AUV operating in dynamic environments. It is primarily an applied research study with the ultimate goal of implementing the proposed path planner in an actual AUV. The research objectives can be summarised as follows:

- Develop a path planner that generates time-optimal paths to improve the efficiency and performance of an AUV operating in complex ocean environments.
- Devise an online path planning scheme that considers the spatiotemporal variability of ocean environments, collision avoidance, and constraints imposed by missions and vehicles.
- Evaluate the effectiveness and scalability of the path planner for AUVs of different sizes and sensor configurations through extensive Monte Carlo methods.
- Verify the practicability and performance of the path planner through numerical and experimental studies.

1.3 Novel Aspects

This research provides contributions through the development of an effective online path replanner to improve the performance of an AUV. The novelties of this work are listed as follows (refer to the Statement Regarding Published Work for publication details):

- A novel PSO-based algorithm has been developed to solve a multi-objective AUV path planning problem. The proposed path planner intelligently exploits ocean currents and generates time-optimal paths to improve the performance of an AUV operating in dynamic and unknown ocean environments. The path planner outperformed other existing algorithms based on systematic benchmark studies. The review and development of the novel PSO-based algorithm were published in Paper 1, Paper 2, and Paper 3, while the application of the algorithm for path planning in dynamic and unknown environments was published in Paper 4 and Paper 5.

- An online path replanning scheme that considers the practical concerns for the operation of an AUV in spatiotemporal and unknown ocean environments is proposed. The proposed scheme continuously refines the vehicle path, which is adaptive to the varying environment based on the limited information provided by onboard sensors. The relevant publications are Paper 4 and Paper 5.

- The proposed path replanner has considered the limitations of different sensor configurations and the availability of sensory data. The planner is also scalable for AUVs of different sizes by adjusting its parameters based on the vehicle dimensions. The path replanner demonstrated its scalability, effectiveness and robustness by using Monte Carlo methods. The relevant publications are Paper 4 and Paper 5.

- The performance of the path replanner has been evaluated and verified under stochastic processes in numerical simulations and hardware-in-the-loop tests, which involved the onboard controllers and actuators of an Explorer class AUV. Most existing relevant studies are based on pure numerical simulations only. The evaluation and implementation of the path replanner were published in Paper 5.

1.4 Assumptions and Scope

The assumptions and scope of the undertaken work in this thesis are outlined as follows:

- This study primarily focused on the development of a high-level planning architecture for AUV missions that involve a single vehicle. The proposed algorithm was only developed and tested for optimizing the planning of a typical torpedo-shaped AUV.

- For the operational scenarios in all chapters, ocean current velocity was assumed to be always lower than the advance velocity of an AUV. This assumption was based on the observation that underwater currents are generally in the range of 0.01 – 0.2 m/s and rarely exceed 1.0 m/s (Shanmugam, 2020), while typical torpedo-shaped AUVs can be operated at up to 3.0 m/s.

- In Chapter 2, 3 and 4, the operational scenarios, including obstacles and ocean current fields, were considered static and exactly known. The AUV was assumed to have a constant water-referenced speed.

- In Chapter 5 and 6, the operational scenarios, including obstacles and ocean current fields, were considered dynamic and fully unknown. The AUV acquired environmental information by using simulated sensor feedback. The sensor measurements were assumed to be reliable and noise-free.
- The proposed algorithm did not incorporate fault and situational awareness. The AUV was assumed to be fully functional throughout the operational scenarios in all chapters.

1.5 Thesis Outline

This thesis consists of seven chapters and four appendices. Chapters 2 – 6 are based on peer-reviewed publications, which are modified to fit into the thesis. The full publications can be found in the appendices. The rest of the thesis is organized as follows:

Chapter 2 provides a comprehensive review of existing PSO-based algorithms, including the basic PSO, QPSO and their variants, for their performances in solving the AUV path planning problem. A pre-generative AUV path planner was developed to solve the offline path planning problem of an AUV operating in a turbulent and cluttered ocean environment. The path planner aimed to generate safe and dynamically feasible time-optimal paths that could exploit ocean currents to improve the AUV's performance. Numerical simulations were conducted to benchmark the algorithms' performances. Using the Monte Carlo methods, the algorithms were evaluated for their solution qualities, stabilities, convergence behaviours and computational requirements.

Chapter 3 proposes a novel approach to improve the performance of PSO-based algorithms in solving an AUV path planning problem by using selective hybridization of DE. To analyse the proposed algorithms, an empirical study and a benchmark study based on several non-linear continuous test functions were conducted. The algorithms were applied in an offline AUV path planner to generate a time-optimal path that can guide the AUV towards its target within minimum time. The performances of the proposed algorithms were evaluated and benchmarked against other algorithms based on Monte Carlo methods and Kruskal-Wallis ANOVA tests.

Chapter 4 presents the formulation of an objective function for AUV path planning as constrained optimization to improve the search efficiency of a path planner. The polar

coordinate system and a combination of hard constraints and soft constraints were used to model the search domain of the path planner. To identify the optimal constraint setting, the effect of different constraints on the algorithm performance was thoroughly analysed using extensive Monte Carlo simulations and Kruskal-Wallis tests.

Chapter 5 addresses the path planning problem of an AUV operating in an unknown, dynamic and cluttered ocean environment. It describes the development of an online AUV path planner that employed a novel PSO-based algorithm and a path replanning approach to optimize the AUV mission. The path replanner aimed to continuously generate and refine a time-optimal path for the AUV based on its onboard sensor measurements throughout a mission. Different configurations for the FLS and H-ADCP sensors were considered in the study. Numerical simulations were conducted to assess the performance and robustness of the proposed path replanner under stochastic processes.

Chapter 6 presents the implementation of the proposed path replanner in an AUV system. The implementation was achieved by using an open-source system architecture, MOOS-IvP. The performance of the path replanner was evaluated and verified in hardware-in-the-loop (HIL) tests. The experiment required the path replanner to interact with the onboard controllers and actuators of an Explorer AUV in real time. A variety of sensor configurations and test scenarios were involved in the experiment. Based on the experimental results, the performance and robustness of the path replanner were evaluated.

Chapter 7 concludes the thesis with key findings for each chapter. Suggestions for further developing the presented work are also given.

Chapter 2.

PSO-based Algorithms for AUV Path Planning

This chapter[1] presents a review of various PSO-based algorithms including the basic PSO, QPSO and their variants. The algorithms were evaluated for the performance in solving the offline path planning problem of an AUV operating in a turbulent and cluttered ocean environment that was *a priori* known. The pre-generative path planner aimed to generate a time-optimal path that could exploit ocean currents to enhance the vehicle performance. Path planning scenarios with obstacles and non-uniform ocean currents were simulated in 2D and 3D domains. Extensive Monte Carlo simulations were conducted to evaluate the solution qualities, stabilities, computational loads, and robustness of the algorithms.

2.1 Literature Overview

Population-based optimization algorithms, such as PSO, are suitable for path planning in dynamic environments, where online planning of a vehicle path is required because it can maintain a large pool of solutions that is available at any time during the mission. These solutions can serve as the initial solutions whenever the replanning of a path is needed, thus significantly improving the computational efficiency. Nonetheless, the algorithm may converge at local minimum solutions if the time allowed for path planning is limited, which is often the case in real AUV operations. In recent years, many strategies that modify the PSO algorithm have been proposed to improve its performance in path planning of various autonomous systems. Each of these variants of PSO was claimed to offer different extents

[1] This chapter was modified from the following publication: Lim, H. S., Fan, S., Chin, C. K. H., & Chai, S. (2018). Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner. IEEE OES Autonomous Underwater Vehicle Symposium, Porto, Portugal.

of improvement over the original PSO algorithm. Nonetheless, there is a lack of systematic methods to evaluate the performances of these algorithms. It is crucial to present a study that compares and reviews these algorithms for the required application.

2.1.1 PSO Algorithm

The PSO algorithm is a metaheuristic population-based optimization technique introduced by Eberhart and Kennedy (1995) based on the inspiration from the analogues of cognitive abilities and social interaction in social animals. Due to its high computational efficiency and easy implementation, PSO has been widely applied in various optimization problems. The pioneering works of PSO application in path planning were presented by Foo et al. (2006), Qin et al. (2004), and Saska et al. (2006). Despite being an evolutionary algorithm, PSO does not have conventional evolutionary operators. PSO consists of particles that move within a multidimensional search space to search for the potential solutions, which are represented by the particles' positions. The particles' velocities are iteratively updated by the particle's own experience (cognitive behaviour) and the entire swarm's experience (social behaviour) to vary the particles' positions. In a standard PSO algorithm that consists of N particles with D number of dimensions for solving an objective function F, the position vector of the i^{th} particle at t^{th} iteration can denoted as:

$$X_i^t = \left[\chi_{i,1}^t , \chi_{i,2}^t ,..., \chi_{i,D}^t \right], \quad i \in \{1,2,...,N\} \tag{2.1}$$

The velocity V and position X of the i^{th} particle at $(t+1)^{\text{th}}$ iteration are updated by using the particle's personal best position *pbest* and the swarm's global best position *gbest* according to Equations (2.2) and (2.3). The positions *pbest* and *gbest* can be determined based on the particles' fitness *F(X)* and their personal best fitness *F(pbest)* as shown in Equations (2.4) and (2.5).

$$V_i^{t+1} = w \cdot V_i^t + C_1 \cdot r_1^t \cdot (pbest_i{}^t - X_i^t) + C_2 \cdot r_2^t \cdot (gbest^{\,t} - X_i^t) \tag{2.2}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{2.3}$$

$$pbest_i{}^t = \begin{cases} pbest_i^{t-1}, & \text{if } F(X_i^t) \geq F(pbest_i^{t-1}) \\ X_i^t, & \text{if } F(X_i^t) < F(pbest_i^{t-1}) \end{cases} \tag{2.4}$$

$$gbest^t = \arg\min \left[F(pbest_i^t) \right] \tag{2.5}$$

where the *argmin* function returns the particle position for the minimum *F(pbest)* among all particles. In Equation (2.2), r_1 and r_2 are uniformly distributed random positive numbers that are less than 1.0. C_1 and C_2 denote the acceleration coefficients for cognitive and social components, respectively; they are both set to 2.0 for most applications (Shi & Eberhart, 1999). The inertia weight w is introduced by Shi and Eberhart (1998) for balancing the global exploration and local exploitation abilities of the particles to produce a faster convergence and hence a better algorithm performance. A common strategy is to set w at an initial value of 0.9 and linearly decrease it to 0.4 according to Equation (2.6) as the algorithm iterates (Eberhart & Shi, 2000).

$$w = w_{\max} - \frac{t}{t_{\max}}\left(w_{\max} - w_{\min}\right) \tag{2.6}$$

To confine the particles within the search space, the particle velocity V is usually bound to an interval of $[-V_{max}, V_{max}]$, where the maximum velocity V_{max} is recommended to be 10 – 20% of the range of variables (Eberhart & Shi, 2001). The application of PSO in AUV path planning can be conducted as described in the following pseudocode after selecting the suitable parameters for the algorithm.

Step 1. **Define** the algorithm parameters and ocean environments.

Step 2. **Initialise** a group of candidate paths by generating particles with random positions in Equation (2.1). Set *pbest* to the current particle positions.

Step 3. **While** the stop criteria are not met,

For $t = 1, 2, \ldots, t_{\max}$,

Evaluate particle fitness $F(X_i^t)$ using the objective function F.

Update *pbest* and *gbest* using Equations (2.4) and (2.5) respectively.

Update w, C_1 and C_2 as required.

For each particle $i = 1, 2, \ldots, N$,

Update particle velocity using Equation (2.2).

Update particle position using Equation (2.3).

End for

End for

End while

Step 4. **Output** *gbest* that holds the optimal path when the stop criteria are met.

2.1.2 QPSO Algorithm

Inspired by the mechanics of quantum systems and PSO algorithm, Sun et al. (2004) proposed the QPSO algorithm, in which the particles are assumed to have quantum behaviour. QPSO is the most well-known variant of PSO. Similar to PSO, QPSO has found applications in many fields. Its application in path planning was pioneered by Fu et al. (2009) and Zhang et al. (2011).

In QPSO, the quantum particles are assumed to be attracted to a 1-dimensional delta potential well centred at a local attraction point for each dimension of the particles' positions. In quantum states, the momentum and energy of the particles are characterized by a wave function, and thus the position and velocity update equations of QPSO are different from the traditional update equations in PSO. Based on a statistical interpretation of the wave function, the probability distribution function of the particles' positions can be obtained. The particles' positions can be converted from quantum states to classical states by employing Monte Carlo inverse transformation on the probability distribution function (Sun et al., 2012). Thus, the position of the i^{th} particle can be given by the following equation.

$$X_i^{t+1} = \begin{cases} \delta_i^t + 0.5 \cdot L_i^t \cdot -\ln\left(u_i^t\right), & \text{if } u_i^t \geq 0.5 \\ \delta_i^t - 0.5 \cdot L_i^t \cdot -\ln\left(u_i^t\right), & \text{if } u_i^t < 0.5 \end{cases} \tag{2.7}$$

where u is a random number uniformly distributed in the range of 0 – 1.0, δ is the potential well as given in Equation (2.8), and L is the characteristic length of the δ potential well as defined in Equation (2.9).

$$\delta_i^t = \varphi_i^t \cdot pbest_i^t + \left(1 - \varphi_i^t\right) \cdot gbest^t \tag{2.8}$$

$$L_i^t = 2 \cdot \beta \cdot \left| mbest^t - X_i^t \right| \tag{2.9}$$

where φ in Equation (2.8) is a random number uniformly distributed in the range of 0 – 1.0. In Equation (2.9), β is the contraction-expansion (CE) coefficient, and *mbest* is the mean best position of the swarm. *mbest* is defined as the average of personal best positions of all the particles in the swarm as shown in Equation (2.10).

$$mbest^{t} = \sum_{i=1}^{N} \frac{pbest_i^t}{N} \tag{2.10}$$

Combining Equations (2.7) - (2.10) yields Equation (2.11), which is the position update equation of particles in the QPSO algorithm.

$$X_i^{t+1} = \varphi_i^t \cdot pbest_i^{\,t} + \left(1 - \varphi_i^t\right) \cdot gbest^{\,t} \pm \beta \cdot \left| mbest^{\,t} - X_i^{\,t} \right| \cdot -\ln\left(u_i^t\right) \tag{2.11}$$

When applying the QPSO algorithm. the selection of β is important for tuning the convergence behaviour of the algorithm. As suggested by an empirical study of parameter selection by Sun et al. (2012), a linearly decreasing β from a maximum value $\beta_{\max}$ of 1.0 to a minimum value $\beta_{\min}$ of 0.5 according to Equation (2.12) is suitable for most applications.

$$\beta = \beta_{\max} - \frac{t}{t_{\max}}(\beta_{\max} - \beta_{\min}) \tag{2.12}$$

The application of QPSO in AUV path planning can be conducted as described in the following pseudocode after selecting the suitable parameters for the algorithm.

Step 1. **Define** the algorithm parameters and ocean environments.

Step 2. **Initialise** a group of candidate paths by generating particles with random positions in Equation (2.1). Set *pbest* to the current particle positions.

Step 3. **While** the stop criteria are not met,

For $t = 1, 2, \ldots, t_{\max}$,

Evaluate particle fitness $F(X_i^{\,t})$ using the objective function F.

Update *pbest* and *gbest* using Equations (2.4) and (2.5) respectively.

Update *mbest* using Equation (2.10).

Update β as required.

For each particle $i = 1, 2, \ldots, N$,

Update particle position using Equation (2.11).

End for

End for

End while

Step 4. **Output** *gbest* that holds the optimal path when the stop criteria are met.

2.1.3 Variants of PSO and QPSO

The performance of PSO-based algorithms depends on their global searching ability, resistance to local minima, convergence speed, robustness, etc. Many strategies have been proposed in recent studies to improve algorithm performance. In this section, the algorithms proposed by these studies were classified into several categories based on the approaches used to improve the algorithm performance. These approaches include algorithm parameters control, novel update equations, hybridization with other algorithms, and a combination of multiple approaches.

Algorithm parameters control

PSO-based algorithms have several parameters that can be adjusted based on their applications, such as their swarm size, maximum iteration, and coefficients of the position update equations. Among these parameters, the equation coefficients are the most critical parameters for controlling convergence behaviour and algorithm performance.

The acceleration coefficients C_1 and C_2, and inertia weight w in the update equation of PSO have to be tuned properly to ensure a balance between global exploration and local exploitation of the particles. In addition to the common approach of setting constant C_1 and C_2, and a linearly decreasing w, some studies introduced an adaptive mechanism in controlling these parameters to ensure they can be adjusted adaptively according to the evolutionary state of particles. Zhan et al. (2009) proposed the adaptive PSO (APSO), in which an evolutionary factor was used as an indicator representing the evolutionary state of particles to control the equation coefficients. To determine the evolutionary factor, the mean particle distance d_i of the i^{th} particle to other particles can be calculated using Equation (2.13). Next, the evolutionary factor f_e can be computed using Equation (2.14).

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} \left(x_i^k - x_j^k\right)^2} \tag{2.13}$$

$$f_e = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \quad \in [0,1] \tag{2.14}$$

where $d_{\min}$ and $d_{\max}$ are the minimum and maximum of the mean particle distances respectively, and d_g is the mean particle distance of the global best particle. During the iteration of APSO, f_e decreases from 1 to 0 as the particles move from the global exploration

phase to the local exploitation phase. Thus, the inertia weight w can be calculated from f_e using a sigmoid mapping as shown in Equation (2.15). The acceleration coefficients C_1 and C_2 can be adapted to f_e using Equation (2.16).

$$w = \frac{1}{1+1.5e^{-2.6f_e}} \quad \in [0.4, 0.9] \tag{2.15}$$

$$\begin{aligned} C_1 &= 0.8 + 2e^{-|f_e - 0.5|} \\ C_2 &= 3.2 - 2e^{-|f_e - 0.5|}, \quad \text{where } C_1 + C_2 = 4 \end{aligned} \tag{2.16}$$

Equation (2.15) allows w to be controlled monotonically with f_e so that it is adaptive to the particle evolutionary state. During the exploration phase, a large f_e and hence large w will improve the global search. In the exploitation phase, a small f_e and hence small w will promote the local search of the solution. APSO has been successfully applied in solving a 2D robotic path planning problem by Song et al. (2017), and in solving the motion planning problem of 3-DOF manipulators by Akbarimajd (2014).

For the QPSO algorithm, the only equation coefficient that needs to be controlled is the CE coefficient β. To adapt β with the particle evolutionary state, Wang et al. (2016) proposed the dynamic-weighted QPSO (DWQPSO) to solve an AUV path planning problem. In DWQPSO, β can be controlled adaptively by classifying the particles into three categories according to their fitness values $F(X_i)$. For the classification of particles, the fitness value of the global best particle is denoted as F_{gbest}, and the average fitness value of all the particles is F_{avg}. The mean of the fitness values for particles that are above average (better than F_{avg}) is denoted as F_{good}. The three categories of particles are as follows:

- $F(X_i) \geq F_{avg}$: For particles with fitness values that are higher than the average, *i.e.*, worse than the average of all the particles, the global exploration ability of the particles should be boosted with a higher β. Thus, β_1 is defined as:

$$\beta_1 = 1.5 - \frac{1}{1+1.5 \cdot e^{F_{gbest} - F_{good}}} \tag{2.17}$$

- $F_{good} < F(X_i) < F_{avg}$: For particles with fitness values that are between F_{good} and F_{avg}, the evolution of the particles should have a balance between global exploration and local exploitation. Hence, β_2 uses a linear decreasing model according to Equation (2.12).

- $F(X_i) \leq F_{good}$: For particles with fitness values that are better than F_{good}, the particles should focus on local exploitation to find the optimal solution. Thus, β_3 is updated using Equation (2.18).

$$\beta_3 = (\beta_2 - 0.5) \cdot \left| \frac{F(X_i) - F_{gbest}}{F_{good} - F_{gbest}} \right| \tag{2.18}$$

Novel update equations

Some studies introduced strategies to improve PSO and QPSO algorithms by modifying their position and velocity update equations. To accelerate the convergence of the algorithms, Yang et al. (2011) simplified the update equation in PSO to propose the accelerated PSO. It was successfully applied in developing a fast and simple path planner by Mohamed et al. (2012). The accelerated PSO disregards the personal best positions of particles and focuses on the global best position by using a simpler update equation as shown in Equation (2.19).

$$X_i^{t+1} = (1 - \Upsilon_1) \cdot X_i^t + \Upsilon_1 \cdot gbest^t + \Upsilon_2^t \cdot \left(u_i^t - 0.5\right) \tag{2.19}$$

where Υ_1 is a randomness parameter that is recommended to be 0.5, u_i^t is a uniformly distributed random number with a value ranging from 0 to 1.0, and Υ_2 is another randomness parameter defined by Equation (2.20). In Equation (2.20), $\Upsilon_{2\max}$ is an initial Υ_2 value that is suggested to be 1.0 and c is a control parameter that takes the value of 0.97.

$$\Upsilon_2 = \Upsilon_{2\max} \cdot e^{-ct} \tag{2.20}$$

Another approach of modifying the PSO update equations was adopted by Zhong et al. (2008), who proposed the phase angle-encoded PSO (θ-PSO) by mapping the position and velocity vector in the equation into a phase angle vector. In θ-PSO, the increment of phase angle replaces the increment of velocity and position through a monotonic sinusoidal mapping function showing in Equation (2.21). The phase angle vector θ of the i^{th} particle at $(t+1)^{\text{th}}$ iteration and its increment $\Delta\theta$ are defined by Equations (2.22) and (2.23).

$$X_i^t = \frac{(X_{\max} - X_{\min}) \cdot \sin\theta_i^t + X_{\max} + X_{\min}}{2} \tag{2.21}$$

$$\Delta\theta_i^{t+1} = w \cdot \Delta\theta_i^t + C_1 \cdot r_1^t \left(pbest_i^t - \theta_i^t\right) + C_2 \cdot r_2^t \left(gbest^t - \theta_i^t\right) \tag{2.22}$$

$$\theta_i^{t+1} = \theta_i^t + \Delta\theta_i^{t+1} \quad \in [-0.5\pi, 0.5\pi] \tag{2.23}$$

Inspired by Zhong et al. (2008), the phase angle-encoded QPSO (θ-QPSO) was introduced by Fu et al. (2012), who applied a similar phase angle mapping concept in QPSO. The fundamental difference between θ-PSO and θ-QPSO is that θ-QPSO does not compute for the phase angle increment, but only the phase angle vector θ as shown in Equation (2.24).

$$\theta_i^{t+1} = \varphi_i^t \cdot pbest_i^t + \left(1-\varphi_i^t\right) \cdot gbest^t \pm \beta \cdot \left| mbest^t - \theta_i^t \right| \cdot -\ln\left(u_i^t\right) \ \in [-0.5\pi, 0.5\pi] \tag{2.24}$$

Fu et al. (2012) developed a UAV path planner and compared the performances of θ-PSO and θ-QPSO. The θ-QPSO path planner was found to have better solution quality, higher robustness and faster convergence.

Hybridization

Another effective method for improving algorithm performance is hybridization, in which the beneficial features of other optimization techniques are combined with the PSO or QPSO algorithm. Zhang and Xie (2003) combined differential evolution (DE) with PSO, resulting in a hybrid algorithm known as DEPSO. The DEPSO algorithm can increase swarm diversity and hence promote its searching ability without altering the original particle swarm dynamics. Based on the inspiration from DEPSO, Fu et al. (2013) adopted a similar hybridization concept to propose the DEQPSO algorithm. Fu et al. (2013) successfully applied DEPSO and DEPQSO for UAV path planning. In both DEPSO and DEQPSO, the conventional position update operation is conducted and followed by a successive three-step DE operation, which consists of three genetic operators described as follows.

- Mutation: A mutated donor vector U is first generated by using Equation (2.25).

$$U_i^t = gbest^t + \frac{(pbest_{r1}^t - pbest_{r2}^t) + (pbest_{r3}^t - pbest_{r4}^t)}{2} \tag{2.25}$$

  where r_1, r_2, r_3 and r_4 are randomly selected particle indices that are mutually different, and different from the current index i and the index of the global best particle, *i.e.*, $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i \neq gbest$.

- Crossover: A trial vector T is generated to increase the diversity by conducting crossover between the donor vector and personal best position as shown in (2.26).

$$T_i^t = \left[\tau_{i1}^t, ..., \tau_{ij}^t, ..., \tau_{iD}^t\right]$$
$$\tau_{ij}^t = \begin{cases} u_{ij}^t, & \text{if } r_j \le CR \,||\, j = r \\ pbest_{ij}^t, & \text{if } r_j > CR \,||\, j \ne r \end{cases} \quad (2.26)$$

where CR is the crossover probability with a suggested value of 0.85, r_j is a random number ranging from 0 to 1.0, and r is a random positive integer ranging from 1 to the total number of dimensions, D, contained by the particle.

- Selection: A greedy selection is used to decide whether the trial vector T should replace the current position X in $(t+1)^{th}$ iteration. X and T are compared according to Equation (2.27). X will only be replaced if T has a better fitness value; otherwise, X will be retained. This means the hybridization of DE operation will never deteriorate the solution but only make it better or remain unchanged.

$$X_i^t = \begin{cases} T_i^t, & \text{if } F(T_i^t) < F(X_i^t) \\ X_i^t, & \text{if } F(T_i^t) \ge F(X_i^t) \end{cases} \quad (2.27)$$

Combination of multiple approaches

In some studies, the algorithms were improved by adopting more than one approach. For example, Modares and Sistani (2011) proposed the IPSO-SQP algorithm, in which an improved PSO (IPSO) with an adaptive inertia weight parameter was combined with the Sequential Quadratic Programming (SQP) algorithm. SQP is a general iterative method used for non-linear constrained optimization. It determines the solution starting from a single search point based on the gradient information. SQP is fast and has a strong searching ability for the local optimal solution, although its solution quality is highly dependent on the initial solution estimation. In IPSO-SQP, SQP is performed to accelerate the local exploitation phase of particles. The inertia weight w of IPSO-SQP is controlled adaptively by the evolutionary state of particles. For each particle, w is defined as a function of its personal best fitness as shown in Equation (2.28).

$$w_i^t = \frac{1}{1 + e^{-\alpha F(pbest_i^t)}} \quad (2.28)$$

where α is the inverse of global best fitness in the first iteration as given in Equation (2.29).

$$\alpha = \frac{1}{F(gbest^1)} \quad (2.29)$$

When the change in global best fitness between iterations in the IPSO-SQP algorithm is less than a predefined value, SQP can be triggered by using the global best solution from the IPSO operation as its initial solution. Upon completing the SQP iteration, the final solution is updated using a greedy selection operator, which only allows the solution from SQP to replace the solution from IPSO if the SQP solution is better; otherwise, the IPSO solution is retained. As a result, the combination of IPSO and SQP can produce faster local convergence and make the solution either become better or remain unchanged but never deteriorate. IPSO-SQP was successfully applied in solving the path planning problem of a REMUS AUV by Taleshian and Minagar (2015).

A hybrid improved QPSO algorithm, known as LTQPSO was proposed by Qian et al. (2015). In LTQPSO, the precision and convergence performance of QPSO can be enhanced by using the particle evolutionary rates, swarm dispersion and hybridization. LTQPSO employs novel update equations, in which individual particle evolutionary rates and swarm dispersion are represented as the control parameters of the equations. The local attractor and position update equations in LTQPSO are given by Equations (2.30) and (2.31) respectively.

$$\delta_i^t = gbest^t + ip_i^t \cdot r \cdot \left(pbest_i^t - gbest^t \right) \tag{2.30}$$

$$X_i^{t+1} = \begin{cases} \delta_i^t + \beta \cdot \left| mbest^t - gs^t \cdot X_i^t \right| \cdot -\ln\left(u_i^t\right), & \text{if } u_i^t \geq 0.5 \\ \delta_i^t - \beta \cdot \left| mbest^t - gs^t \cdot X_i^t \right| \cdot -\ln\left(u_i^t\right), & \text{if } u_i^t < 0.5 \end{cases} \tag{2.31}$$

where ip_i^t and gs^t are the dynamic control parameters for particle evolutionary rates and swarm dispersion, respectively. ip_i^t is defined by Equation (2.32), which gives the ratio of the global best fitness to the personal best fitness of particles. gs^t is defined by Equation (2.33), which gives the ratio of the standard deviation of personal best positions recorded by all particles to the standard deviation of the particles' current positions.

$$ip_i^t = \frac{F\left(gbest^t\right)}{F\left(pbest_i^t\right)} \quad \in [0,1] \tag{2.32}$$

$$\begin{aligned} gs^t &= \left[gs_{i,1}^t, gs_{i,2}^t, \ldots, gs_{i,D}^t \right] \\ &= \left[\frac{\sigma\left(pbest_{i,1}^t\right)}{\sigma\left(X_{i,1}^t\right)}, \frac{\sigma\left(pbest_{i,2}^t\right)}{\sigma\left(X_{i,2}^t\right)}, \ldots, \frac{\sigma\left(pbest_{i,D}^t\right)}{\sigma\left(X_{i,D}^t\right)} \right] \end{aligned} \tag{2.33}$$

For each iteration in LTQPSO, a natural selection operator is performed after the QPSO operation. The operator sorts the particles according to their personal best fitness values and replaces the worst-performing particles with the best-performing particles. A natural selection parameter ς with a suggested value of 2 is used to control the number of particles N_s that will be replaced.

$$N_s = \frac{N}{\varsigma} \tag{2.34}$$

The natural selection operator increases the evolutionary rate of the entire swarm by eliminating the least desirable solutions to produce a faster global convergence. Xue et al. (2017) applied LTQPSO for path planning in a complex 2D environment.

2.2 Problem Formulation

In this thesis, the primary objective of the AUV path planner was to solve a multi-objective non-linear optimization problem, in which the Pareto-optimal path for the AUV to travel towards a target location through ocean environments was required to be determined. In addition, the path planner aimed to generate a time-optimal path that could exploit ocean currents to improve vehicle performance. This section describes the formulation of the optimization problem.

2.2.1 Path Formulation

A potential path of an AUV can comprise a series of nodes along the path from a starting point to an end point (target). Controlling and optimizing the coordinates of the path nodes can produce an optimized path for the AUV. Neither the starting point nor the end point of a path should be involved in the optimization process because all potential paths share the same start and end locations.

In PSO-based path planners, a potential path solution can be modelled as a particle in the swarm. The swarm population can be denoted by a matrix $X = [X_1, X_2, \dots, X_N]^{\mathrm{T}}$, where X_i is the position vector of particles and N is the total number of particles in the swarm. The entries of a particle's position vector represent the coordinates of the path nodes. Assuming every path consists of n+2 nodes including the starting and end points, the number of nodes involved in the optimization process is n. To record the Cartesian coordinates of n node(s) in a 2D Euclidean plane, the position vector of a particle requires 2n dimensions for x and

y coordinates. For n node(s) in a 3D Euclidean space, a particle requires $3n$ dimensions to include an additional n dimension(s) for z coordinate. The respective position vectors of the i^{th} particle at t^{th} iteration for the 2D and 3D problems can be written as Equations (2.35) and (2.36).

$$X_i^t = \left[\chi_{i,1}^t, ..., \chi_{i,n}^t, \chi_{i,n+1}^t, ..., \chi_{i,2n}^t \right], \quad i \in \{1,2,...,N\} \tag{2.35}$$

$$X_i^t = \left[\chi_{i,1}^t, ..., \chi_{i,n}^t, \chi_{i,n+1}^t, ..., \chi_{i,3n}^t \right], \quad i \in \{1,2,...,N\} \tag{2.36}$$

The path nodes, including the starting and end points, can be connected to form an AUV path by using B-splines, which are parametric curves generated from a series of connected piecewise polynomials. B-spline is suitable for this application because it offers the following properties:

- It can produce a practical path shape without unnecessary wiggling segments, hence reducing the control actions required by an AUV to follow the path.
- It can maintain the continuity of its second derivative (C^2) and curvature function (G^2). This enhances the path following performance of an AUV.
- It can offer local control for path alteration without loss of continuity. This allows the effect of path nodes relocation to be localized to the adjacent path segments only.

The path nodes can serve as the control points for B-splines according to the curve function in Equation (2.37) , which gives the output vector $P(\kappa)$ representing a B-spline curve with $j+1$ order in the form of discretised waypoints. If the total number of control points is $n+2$, the total number of piecewise polynomials is one less than the number of control points, which is $n+1$.

$$P(\kappa) = \sum_{i=0}^{n+1} x_i \, B_{i,j}(\kappa), \qquad i \in \{0,1,2,...,n+1\} \tag{2.37}$$

where x_i denotes the control points, κ is the non-decreasing knot sequence given by a knot vector $\kappa = [\kappa_0, \ldots, \kappa_i, \ldots, \kappa_{n+j+2}]$, and $B_{i,j}(u)$ represents the piecewise polynomial basis functions of j degree defined by Cox de Boor recursion (Piegl & Tiller, 2012) as follows.

$$B_{i,0}(\kappa) = \begin{cases} 1, & \text{if } \kappa_i \le \kappa \le \kappa_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.38}$$

$$B_{i,j}(\kappa) = \frac{\kappa - \kappa_i}{\kappa_{i+j} - \kappa_i} B_{i,j-1}(\kappa) + \frac{\kappa_{i+j+1} - \kappa}{\kappa_{i+j+1} - \kappa_{i+1}} B_{i+1,j-1}(\kappa) \tag{2.39}$$

The continuity of a B-spline is fully dependent on the basis functions. Hence, the control points, *i.e.*, path nodes can be adjusted during the path optimization process without affecting the continuity of B-splines.

2.2.2 Objective Functions

The application of PSO-based algorithms in an optimization problem requires the development of suitable objective functions to evaluate the fitness of particles based on their respective solutions. Due to the high computational efficiency of PSO-based algorithms, the evaluation of particle fitness using objective functions normally contributes to the majority of algorithm runtime (Sun et al., 2012). Objective functions should be developed in accordance with the optimization criteria of the problem. To provide an accurate fitness representation model for finding the optimal solution, the developed function must closely resemble the physical conditions of the problem space. The optimization criteria of a multi-objective AUV path planning problem are:

- Minimum path length or travel time required to reach a target.
- Minimum exposure to threats, *i.e.*, obstacle avoidance.
- Compliance with the physical motion limitations of an AUV.

A trade-off between these criteria should be established by using multiple objective functions because the optimum of all criteria does not necessarily coincide. In this work, the path planner was modelled using an aggregate function with equal weights assigned to the underlying objective functions F_k. The optimal path solution X^* can be given by the function in Equation (2.40). Path planning is a minimization problem that requires the vehicle's path length or travel time to be minimized. Therefore, an optimal solution should have the lowest cost/fitness.

$$X^* = \arg\min \sum_{k=1}^{K} F_k(X_i) \tag{2.40}$$

where k refers to different objective functions and K is the total number of functions in the path planning problem.

Travel time optimality

The first objective function F_1 was developed to measure the fitness of a path based on its length or time to travel on the path. This study focused on finding a time-optimal path that could achieve a minimum travel time by exploiting favourable currents to assist AUV motions while avoiding less favourable currents. For this purpose, a travel-time-based objective function was employed. By taking into consideration the effect of ocean currents, the objective function F_1 can allow the path planner to adapt its solutions to currents and generate time-optimal paths, which could guide an AUV to its target within a minimum time. This objective function assumed that the current velocity is always lower than the advance velocity of an AUV. This assumption was based on the observation that underwater currents are generally in the range of 0.01 – 0.2 m/s and rarely exceed 1.0 m/s (Shanmugam, 2020), while typical torpedo-shaped AUVs can be operated at up to 3.0 m/s (e.g., the REMUS 100 AUV has a maximum operational speed of 2.6 m/s).

A given path X_i can be represented as a sequence of discretised waypoints $P = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$, where P is the output from the B-spline function and m is the total number of discretised waypoints. The travel time fitness $F_1(X_i)$ of a path can be measured by finding the sum of discrete time required to travel on every small path segment that connects consecutive waypoints in P as shown in Equation (2.41).

$$F_1(X_i) = \sum_{j=1}^{m-1} \frac{\left\| \overrightarrow{p_{i,j} p_{i,j+1}} \right\|}{\left| V_g \right|}, \quad j \in \{1, 2, ..., m-1\} \tag{2.41}$$

where the numerator gives the Euclidean distance between two consecutive waypoints. V_g denotes the resultant ground-referenced velocity of an AUV, which is the resultant AUV velocity under the effect of surrounding ocean currents. The effect of currents on the AUV can be obtained by projecting the velocity vector of currents V_c in the direction of the AUV water-referenced velocity V_a, which is essentially the direction of the path vector. Thus, V_g can be given by summing V_a and the effect of V_c as shown in Equation (2.42).

$$V_g = V_a + \frac{V_c \cdot \overrightarrow{p_{i,j} p_{i,j+1}}}{\left\| \overrightarrow{p_{i,j} p_{i,j+1}} \right\|} \tag{2.42}$$

Obstacle avoidance

The second objective function was designed as a penalty function for achieving obstacle avoidance. The penalty function F_2 can measure a path's exposure to threats/obstacles in terms of threat cost $F_2(X_i)$. All the obstacles were modelled as ellipses (or circles if the major axis and minor axis are equal) in the 2D problem space, and as ellipsoids (or spheres if all the principal axes are equal) in the 3D space with their principal axes aligned with the coordinate axes. An intuitive method of measuring the threat cost is to calculate the Euclidean distances of all the discretised waypoints to the centres of obstacles by using Equation (2.43) and penalise the cost if the distances are smaller than the semi-major axes (semi-principal axes) of the obstacles.

$$d_{obs} = \left\| \overrightarrow{p_{i,j}\, O_{c,h}} \right\| \qquad h \in \{1,2,...,H\} \tag{2.43}$$

where O_c denotes the centre of an obstacle, h refers to different obstacles and H is the total number of obstacles in the problem space. However, the accuracy of this threat cost measurement method is fully dependent on the fineness of a path, *i.e.*, the number of discretised waypoints generated by the B-spline function. The threat cost can be inaccurate when the distance between two consecutive waypoints is greater than the minor axis of an obstacle. To ensure the threat cost can be measured accurately using this method, a finer path with a high number of discretised waypoints must be used. This can lead to a higher computational requirement for path generation and evaluation.

To ensure the accuracy of threat cost evaluation, this study employed a method that can measure the threat cost based on the intersection between path segments and obstacles. The intersection-based method has fineness-independent accuracy, meaning that the path fineness can be lowered without compromising the accuracy of threat cost. The threat cost for 2D and 3D problems can be measured by using the same approach, except that the dimension reduction of 2D problems reduced the number of variables and hence simplified the computation. Assuming an obstacle h in 3D problem space with a centre $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and semi-principal axes $O_{r,h} = (O_{rx}, O_{ry}, O_{rz})$, its parametric equation can be expressed in Equation (2.44). The parametric equation with a parameter s for a path segment that connects two consecutive waypoints $p_{i,j} = (x_1, y_1, z_1)$ and $p_{i,j+1} = (x_2, y_2, z_2)$ can be written as Equation (2.45).

$$\left(\frac{x - O_{cx}}{O_{rx}}\right)^2 + \left(\frac{y - O_{cy}}{O_{ry}}\right)^2 + \left(\frac{z - O_{cz}}{O_{rz}}\right)^2 = 1 \tag{2.44}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + s \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{2.45}$$

An alert distance d_{alert} was used to check whether the path was safe from the obstacle. It can be given by Equation (2.46), in which max($O_{r,h}$) refers to the longest semi-principal axis of the obstacle and max($\| \overrightarrow{p_{i,j}\ p_{i,j+1}} \|$) is the length of the longest path segment.

$$d_{alert} = \max\left(O_{r,h}\right) + \max\left(\left\| \overrightarrow{p_{i,j}\ p_{i,j+1}} \right\|\right) \tag{2.46}$$

When d_{obs} was greater than d_{alert}, the obstacle can be safely ignored because it was too far from the path. If an obstacle came within the alert distance of a path segment, the function should check for the intersection between the segment and the obstacle. The intersection can be determined by substituting Equation (2.45) into Equation (2.44) to yield the following equations, which is expressed in terms of s.

$$A s^2 + B s + C = 0 \tag{2.47}$$

where A, B and C represent Equations (2.48), (2.49) and (2.50), respectively.

$$A = \frac{O_{ry}{}^2 O_{rz}{}^2 (x_2 - x_1) + O_{rx}{}^2 O_{rz}{}^2 (y_2 - y_1) + O_{rx}{}^2 O_{ry}{}^2 (z_2 - z_1)}{O_{rx}{}^2 O_{ry}{}^2 O_{rz}{}^2} \tag{2.48}$$

$$B = \frac{(O_{cx} - x_1)(x_1 - x_2)}{0.5\, O_{rx}{}^2} + \frac{(O_{cy} - y_1)(y_1 - y_2)}{0.5\, O_{ry}{}^2} + \frac{(O_{cz} - z_1)(z_1 - z_2)}{0.5\, O_{rz}{}^2} \tag{2.49}$$

$$C = \frac{(O_{cx} - x)^2}{O_{rx}{}^2} + \frac{(O_{cy} - y)^2}{O_{ry}{}^2} + \frac{(O_{cz} - z)^2}{O_{rz}{}^2} - 1 \tag{2.50}$$

The intersection of the path with the obstacle can be evaluated by obtaining the discriminant Δ of Equation (2.47) according to Equation (2.51).

$$\Delta = B^2 - 4AC \tag{2.51}$$

A buffer distance was added to the principal axes of every obstacle to expand the obstacle space. The buffer distance considered the dimension of an AUV and ensured the vehicle kept a safe distance from actual obstacles. Thus, the AUV will not conflict with obstacles even when $\Delta = 0$, *i.e.*, the path is tangent to the buffered obstacle. The setting of buffer distance can be adjusted based on the vehicle dimension. When $\Delta > 0$, the path can be threatened by the obstacle if the roots s_1 and s_2 given by Equation (2.52) are within the range of [0,1].

$$s_1,\ s_2 = \frac{-B \pm \sqrt{D}}{2A} \tag{2.52}$$

The intersection points, S_1 and S_2, can be determined by solving Equation (2.45) using s_1 and s_2. The threat cost $F_{2,h}(p_{i,j})$ was proportional to the length of the path segment contained in the buffered obstacle space as given in Equation (2.53). The total threat cost $F_2(X_i)$ of a solution X_i can be obtained by using Equation (2.54).

$$F_{2,h}(p_{i,j}) = \frac{\| \overrightarrow{S_1\ S_2} \|}{2 \cdot \max(O_{r,h})} \tag{2.53}$$

$$F_2(X_i) = \sum_{h=1}^{H} \sum_{j=1}^{m-1} F_{2,h}(p_{i,j}) \tag{2.54}$$

The following pseudocode describes the procedure for finding the threat cost $F_2(X_i)$.

For each threat $h = 1, 2, \ldots, H$,
  Use Equation (2.43) to find d_{obs}, which is the distance between $p_{i,j}$ and $O_{c,h}$.
  For each discretised waypoint $j = 1, \ldots, m\text{-}1$,
    If $d_{obs} >$ **max**$(O_{r,h})$ + **max**$(\| \overrightarrow{p_{i,j}\ p_{i,j+1}} \|)$
      $F_{2,h}(p_{i,j}) = 0$
    Else
      Find Δ using Equation (2.51).
      If $\Delta \leq 0$
        $F_{2,h}(p_{i,j}) = 0$
      Else
        Find s_1 and s_2 using Equation (2.52).
        If $0 \leq s_1 \leq 1$ **and** $0 \leq s_2 \leq 1$
          Find the intersection points S_1 and S_2 by solving Equation (2.45).
          Calculate the threat cost using Equation (2.53).
        Else
          $F_{2,h}(p_{i,j}) = 0$
        End if
      End if
    End if
  End for
End for
Calculate the total threat cost using Equation (2.54).

Vehicle motion limitations

A feasible vehicle path must comply with the physical motion limitations of an AUV, which should include its yaw (turning) and pitch motions. The following objective functions were developed to check the compliance of path solutions with respect to these limitations. The fitness of a path should be penalised if any of the limitations are violated. For the yaw limitation, the turning angle of a path can be measured and checked against the maximum allowable turning angle $\Psi_{\max}$ of the AUV. Considering two consecutive path segments that consist of three waypoints $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j+2}$ (refer to Figure 2.1), the turning angle Ψ can be obtained from the cosine function as shown in Equation (2.55). The first part of the function is the scalar projection of the $p_{i,j}$, $p_{i,j+1}$ segment on the $p_{i,j+1}$, $p_{i,j+2}$ segment in the x-y plane, while the second part is the length of the $p_{i,j+1}$, $p_{i,j+2}$ segment in the x-y plane.

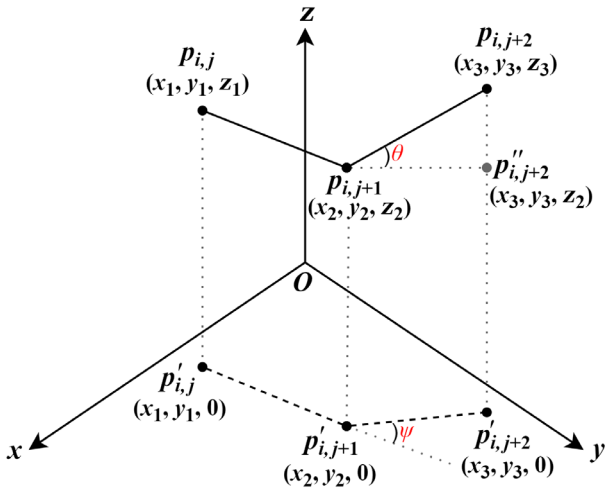


Figure 2.1: Yaw and pitch angles of a path.

$$\psi_j = \cos^{-1}\left(\frac{\overrightarrow{p'_{i,j}\,p'_{i,j+1}} \cdot \overrightarrow{p'_{i,j+1}\,p'_{i,j+2}}}{\left\| \overrightarrow{p'_{i,j}\,p'_{i,j+1}} \right\|} \cdot \frac{1}{\left\| \overrightarrow{p'_{i,j+1}\,p'_{i,j+2}} \right\|} \right) \tag{2.55}$$

The penalty $F_3(X_i)$ for violating the yaw limitation can be obtained from the turning angle by using Equation (2.56).

$$F_3(X_i) = \sum_{j=1}^{m-1} F_3(p_{i,j})$$

$$F_3(p_{i,j}) = \begin{cases} 0 & , \text{ if } |\psi_j| \le \psi_{\max} \\ 1 - \dfrac{\pi - |\psi_j|}{\pi - \psi_{\max}} & , \text{ if } |\psi_j| > \psi_{\max} \end{cases} \tag{2.56}$$

The maximum allowable turning angle ψ_{max} of an AUV path was estimated from the vehicle's minimum turning radius R_{curv}, which can be converted into a degree of curvature θ_{curv}. Referring to Figure 2.2, θ_{curv} is equivalent to ψ_{max} and can be approximately calculated from two consecutive path segments with the minimum distance d_{pp} according to Equation (2.57). This approximation underestimated the arc length of the turning radius, giving a lower θ_{curv} and hence a lower and more conservative estimation of ψ_{max}, which is desirable for a safe and feasible path.

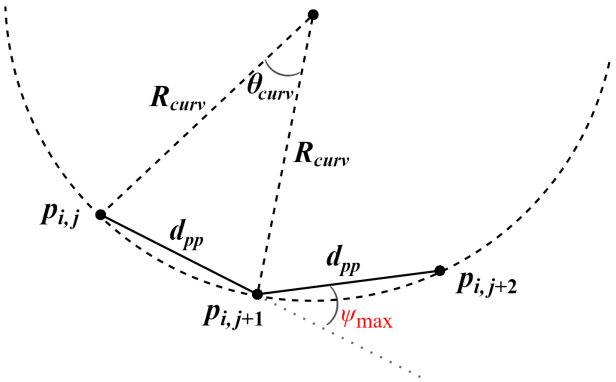


Figure 2.2: Estimation of maximum turning angle from minimum turning radius.

$$\psi_{max} = \theta_{curv} \approx \frac{d_{pp}}{R_{curv}} \tag{2.57}$$

where θ_{curv} must be in radians, and d_{pp} is the minimum distance between the waypoints of a generated path.

For the pitch motion, the pitch angle θ and change in pitch $\Delta\theta$ of the AUV at any point should not exceed their respective maximum values, θ_{max} and $\Delta\theta_{max}$. Referring to Figure 2.1, θ can be determined using a basic tangent function as shown in Equation (2.58). Next, $\Delta\theta$ can be calculated by using Equation (2.59).

$$\theta_j = \tan^{-1}\left[\frac{\left\| p_{i,j+2}\overrightarrow{p''_{i,j+2}} \right\|}{\left\| p_{i,j+1}\overrightarrow{p''_{i,j+2}} \right\|}\right] \tag{2.58}$$

$$\Delta\theta_j = \theta_{j+1} - \theta_j \tag{2.59}$$

From the calculated pitch, the penalty $F_4(X_i)$ for violating θ_{max} and the penalty $F_5(X_i)$ for $\Delta\theta_{max}$ can be obtained by using Equations (2.60) and (2.61), respectively.

$$F_4(X_i) = \sum_{j=1}^{m-1} F_4(p_{i,j})$$

$$F_4(p_{i,j}) = \begin{cases} 0 & , \text{ if } |\theta_j| \le \theta_{\max} \\ 1 - \dfrac{0.5\pi - |\theta_j|}{0.5\pi - \theta_{\max}} & , \text{ if } |\theta_j| > \theta_{\max} \end{cases} \tag{2.60}$$

$$F_5(X_i) = \sum_{j=1}^{m-2} F_5(p_{i,j})$$

$$F_5(p_{i,j}) = \begin{cases} 0 & , \text{ if } |\Delta\theta_j| \le \Delta\theta_{\max} \\ 1 - \dfrac{0.5\pi - |\Delta\theta_j|}{0.5\pi - \Delta\theta_{\max}} & , \text{ if } |\Delta\theta_j| > \Delta\theta_{\max} \end{cases} \tag{2.61}$$

2.3 Simulation Setup

Numerical simulations were conducted to benchmark the performance of different PSO-based algorithms for the path planning of an AUV. The algorithms were applied in an offline path planning scenario using a pre-generative scheme. The benchmark study can provide a baseline to evaluate the algorithms' performances in demonstrating an AUV path planner's minimum capability, which can also reflect their performances when applied in an online path planner. The following assumptions were made for the offline path planning simulation.

- The AUV has constant water-referenced velocity and thus constant thrust.
- The ocean environment was *a priori* known with negligible uncertainties. The predicted ocean currents and obstacle locations were exact, static and time-invariant.
- The AUV can follow the generated path accurately.

The benchmark study was based on the Monte Carlo method with 1000 simulation runs. The simulations were conducted in 2D domains followed by 3D domains. The problem spaces of the simulations were assumed to be an ocean field that is 50×50 square metres for 2D and 50×50×50 cubic metres for 3D. Non-uniform ocean currents and static obstacles of different sizes were present in the problem space. The ocean currents were generated based on the field data obtained at Beauty Point, Tasmania, Australia. The field data were acquired by using the ADCP sensors of the UTAS Explorer AUV "*nupiri muka*" during one of the vehicle's open water trials for the preparation of its Antarctic expedition (Pyper, 2018).

During the simulation, the AUV was required to travel with a preset water-referenced velocity of 1.5 m/s. The buffer distance for obstacle avoidance was set to 1 metre. The minimum turning radius was set to 8.1 metres, while the angles θ_{max} and $\Delta\theta_{max}$ were set to 45° and 10° respectively. These settings were based on the properties of the REMUS 100 AUV, which has a length of 1.7 metres, a diameter of 0.19 metres and a dry weight of 36 kg (refer to Appendix A for more detailed specifications).

In each simulation run, the maximum number of algorithm iterations was set to 100 with a predefined stopping threshold. The algorithm was allowed to iterate up to a maximum number of 100, but it was stopped whenever the difference of solutions between successive iterations was less than the preset threshold. The population size of all the algorithms was set to 150 particles, with each particle consists of 4 path nodes, meaning each particle has 8 dimensions for the 2D scenarios and 12 dimensions for the 3D scenarios. For all the PSO-based algorithms, the inertia weight w was set to be linearly decreasing from 0.9 to 0.4, and the acceleration coefficients C_1 and C_2 were both set to 2.0 as suggested by Shi and Eberhart (1999). The CE coefficient β of all the QPSO-based algorithms was set to be linearly varying from 1.0 to 0.5 as suggested by Sun et al. (2012). Other algorithm parameters for the variants of PSO and QPSO were set to be their respective suggested values, which were discussed in Section 2.1.3. The simulations were conducted on a machine with Intel Core i5-6300U CPU @ 2.4GHz with 8GB RAM.

2.4 Benchmark of PSO-based Path Planners

Figure 2.3 depicts the Pareto-optimal path solutions generated by the PSO-based algorithms in the Monte Carlo simulations under 2D and 3D scenarios. In both scenarios, the AUV was required to travel from the starting point (green square) to the target (pink star) without running into obstacles, while trying to take advantage of the favourable current to assist the AUV motion. Using a colour bar in the 2D results of Figure 2.3, the blue regions indicate favourable ocean currents, while the red regions show less favourable currents. The favourable currents were defined as positive and flowing towards the top right corner, hence pushing the vehicle from its starting position towards the target. On the other hand, the less favourable currents were defined as negative and flowing in the opposite direction (towards the bottom left corner).

In both results, the solid sections of the paths indicate that the favourable ocean currents had positive effects on the AUV motions, whereas the dotted sections suggest otherwise. It can be observed that the majority of the generated paths were able to exploit the favourable currents and avoid the less favourable currents to improve vehicle efficiency.

The detailed results of the 2D and 3D simulations are presented in the next two sections, in which the performances of the algorithms were compared based on their solution qualities, stabilities, convergence behaviours, and computational requirements. These properties were evaluated by studying the fitness values of the solutions obtained and the runtime required to obtain the solutions.

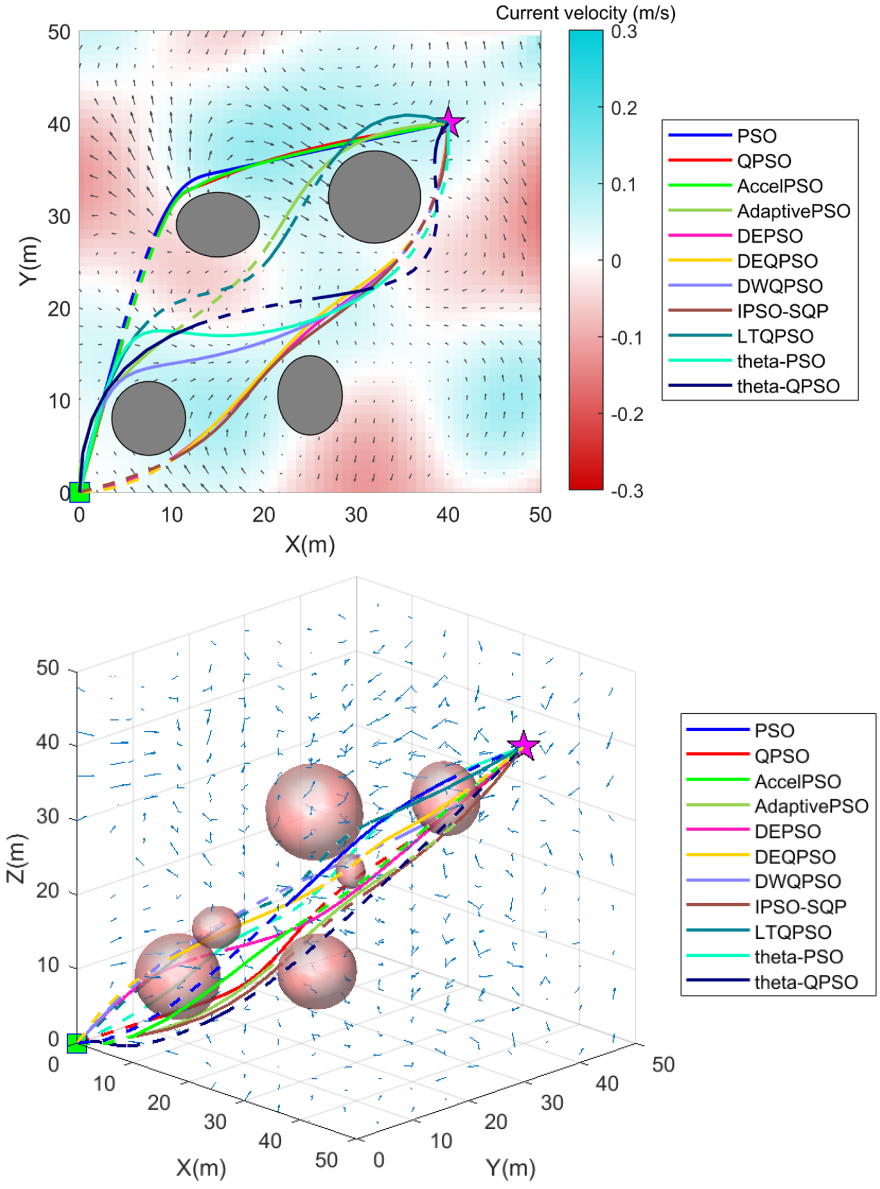


Figure 2.3: Pareto-optimal path solutions for 2D (top) and 3D (bottom) scenarios.

2.4.1 Convergence Behaviours

The convergence behaviours of the algorithms under 2D and 3D scenarios are compared in Figure 2.4. The convergence speed of the algorithm can be given by the minimum number of iterations required for the algorithm to converge at an optimal or sub-optimal solution. It can be observed in the graphs that the convergence speeds of all the algorithms significantly decreased when the dimensionality of the problem increases from 2D to 3D. Based on the convergence curves, DEPSO and DEQPSO outperformed other algorithms with similar performance in both scenarios; the two algorithms achieved the fastest convergence and the global convergence with the lowest fitness. In addition, APSO and IPSO-SQP were also able to offer faster and better convergence than the standard PSO and QPSO under both scenarios. The convergence curves show that θ-PSO and θ-QPSO performed poorly, especially in 3D.

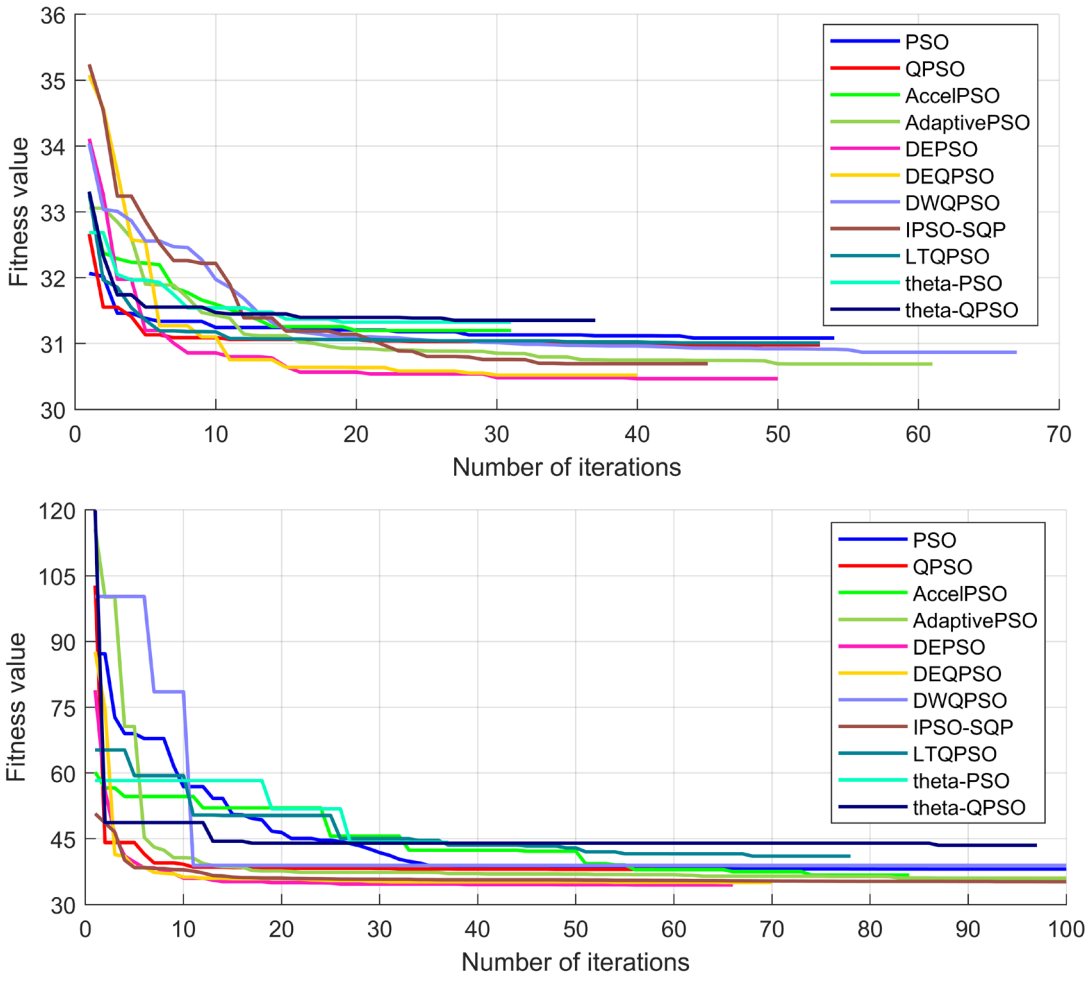


Figure 2.4: Convergence curves for 2D (top) and 3D (bottom) scenarios.

2.4.2 Solution Qualities and Computational Loads

Prior to further evaluating the algorithm performance, the normality of the 2D and 3D Monte Carlo simulation results was examined by using the Shapiro-Wilk test with a significance level of 0.05. The normality test revealed that the data was not normally distributed. Hence, the median and interquartile range were used as the indicators for solution quality and stability. The boxplots in Figure 2.5 and Figure 2.6 show the fitness values of the solutions obtained and the runtime required to obtain the solutions, respectively. The fitness value of a solution simply represents the time required for the AUV to reach the target by travelling on the generated path. Therefore, a lower fitness value indicates a higher solution quality and hence a stronger searching ability.

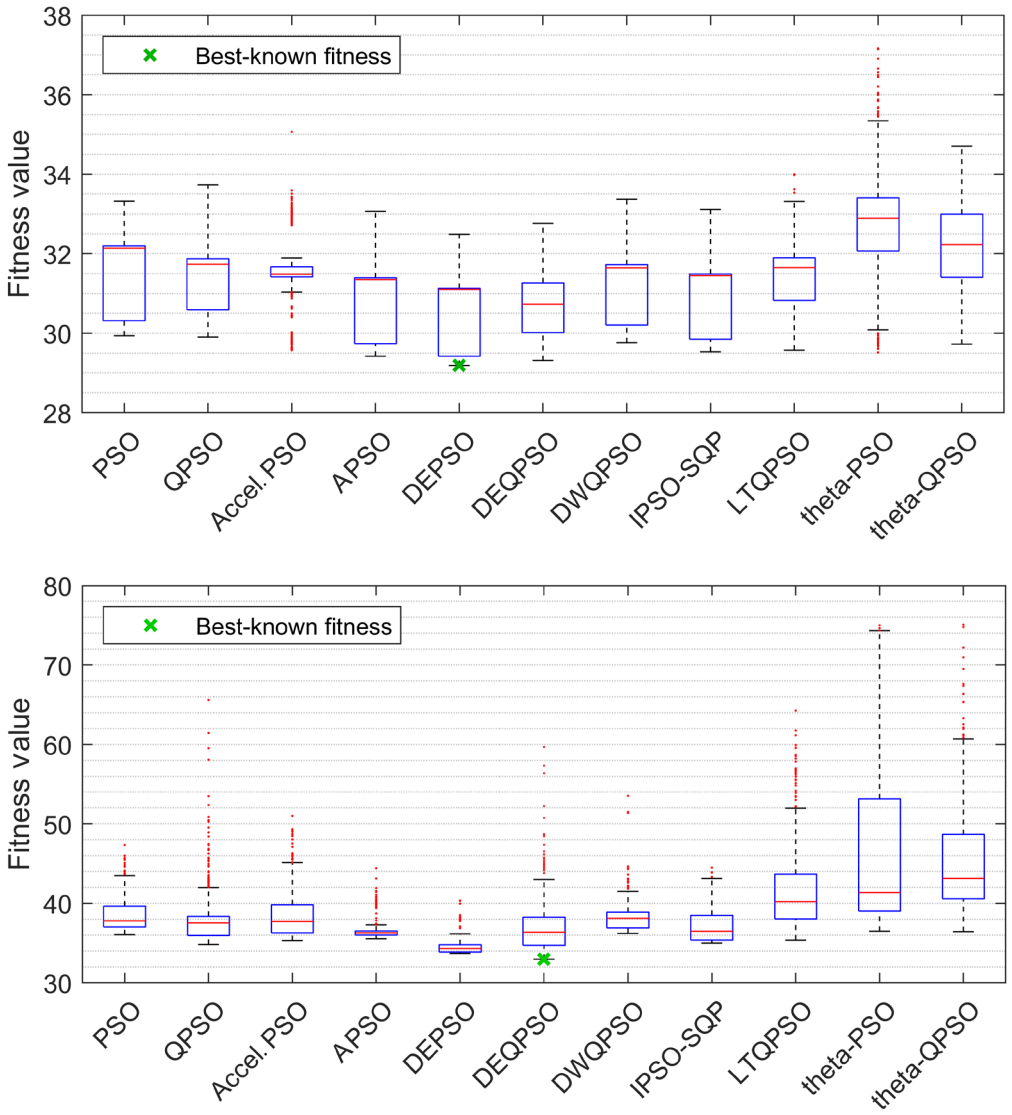


Figure 2.5: Fitness values obtained in 2D (top) and 3D (bottom) scenarios.

In the boxplots, the blue box shows the interquartile range of the data while the upper and lower ends of the box indicate the 25th to 75th percentile. The median is indicated by the red horizontal line inside the box. The acceptable data range is indicated by the black whisker, and the outliers are represented by red dots. For the boxplots of fitness values, the lower end of each whisker identifies the individual best fitness obtained by each algorithm over the 1000-run simulation, while the green cross sign represents the best-known (lowest) fitness value among all the algorithms in the simulations. The acceptable data range, percentile range and outliers are indicators for the stabilities of the algorithm performance, while the medians give information about the solution qualities and searching abilities of the algorithms.

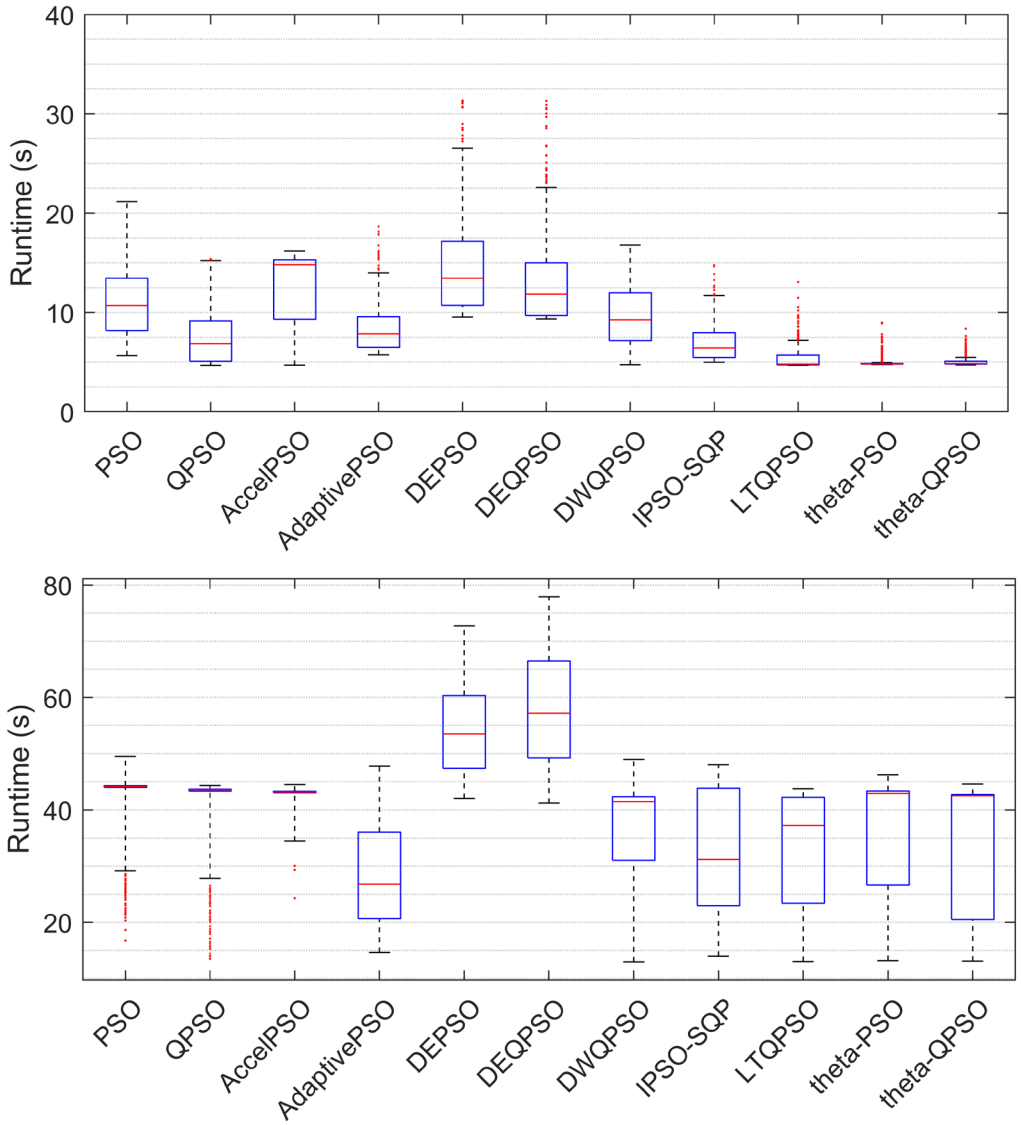


Figure 2.6: Algorithm runtime for 2D (top) and 3D (bottom) scenarios.

Based on the simulation results, the two DE-hybridized algorithms outperformed almost every other algorithm in terms of solution qualities in both 2D and 3D scenarios. As shown in Figure 2.5, DEQPSO achieved the lowest median in 2D and the second-lowest median in 3D. DEQPSO's individual best fitness was the second best-known fitness in 2D and the best-known fitness in 3D. Following closely the performance of DEQPSO, the median fitness of DEPSO was the second-lowest in 2D and the lowest in 3D. DEPSO also achieved the best-known fitness in 2D and the second best-known fitness in 3D. The hybridization of DE operation in the PSO-based algorithms offered up to a 9% improvement in the searching ability of the algorithms (in terms of fitness values when compared to the standard PSO).

However, as shown in Figure 2.6, DEPSO and DEQPSO required significantly higher runtime compared to other algorithms, and the increase in runtime was even more drastic when the problem space increased to 3D. This was caused by the greedy selection operator, which required an additional fitness evaluation step for every particle to compare its fitness with the fitness of its trial vector during the DE operation. This means the fitness of all particles needed to be evaluated twice for every iteration. The greedy selection operator can drastically increase the computational requirements of the algorithms because the evaluation of particle fitness contributes to most of the computational time in PSO-based algorithms.

APSO and IPSO-SQP were also able to offer excellent performances; both generated higher solution qualities than the standard PSO and QPSO. In the 2D scenario, APSO had the third-lowest median fitness, while IPSO-SQP had the fourth-lowest median. In the 3D scenario, both APSO and IPSO-SQP had median fitness that was close to the median of DEQPSO. More importantly, the two algorithms showed a good balance between solution quality and runtime. The medians of their runtime were lower than most of the tested algorithms, indicating their high efficiency in solving the path planning problem.

Although DWQPSO achieved a comparable median fitness, its runtime was significantly higher than the average. The accelerated PSO and LTQPSO did not offer significant performance improvement over PSO and QPSO in terms of solution quality, although LTQPSO required less runtime. θ-PSO and θ-QPSO were found to have inferior performance based on their poor median fitness. The extremely low runtime of the two algorithms in 2D indicates that they were prone to be trapped by local minima. In the 3D scenario, the two algorithms had significantly higher median fitness and interquartile ranges, indicating their inferior and unstable performances.

2.5 Chapter Summary

The performances of various PSO-based algorithms, including PSO, QPSO and their variants, has been reviewed for the application of AUV path planning in this chapter. The variants of PSO and QPSO were studied and classified into several categories based on the approaches used to improve the algorithm performance. A pre-generative AUV path planner was developed to solve an offline path planning problem by using the PSO-based algorithms. Numerical simulations were conducted to benchmark the algorithms' performances in accomplishing an AUV path planner's minimum capability. The path planning simulations were conducted in a turbulent and cluttered ocean environment under 2D and 3D scenarios.

Based on the Monte Carlo methods, the algorithms were evaluated for their solution qualities, stabilities, convergence behaviours and computational requirements. The simulation results showed that DEPSO and DEQPSO outperformed other tested algorithms with equivalently excellent performance in terms of solution qualities, stabilities and convergence behaviours. The hybridization of DE operation offers up to a 9% improvement in the searching ability of particles when compared to the standard PSO algorithm. However, the computational requirement of the DE-hybridized algorithms was found to be higher due to the greedy selection operator, which required the particles to undergo twice the evaluation of fitness.

APSO and IPSO-SQP were also found to have extraordinary performances in both 2D and 3D scenarios. They were able to achieve a balance between solution qualities and computational requirements, with their solution qualities slightly lower than the DE-hybridized algorithms. Most importantly, DEPSO, DEQPSO, APSO and IPSO-SQP were proven to have higher algorithm performances than other tested PSO-based algorithms. They were capable of generating safe and feasible AUV paths that can achieve obstacle avoidance, comply with the AUV motion limitations and exploit ocean currents in *a priori* known environments.

The main purpose of the review in this chapter is to act as a baseline for developing an online path planner by using a PSO-based algorithm. The benchmark study successfully identified the strengths and weaknesses of different PSO-based algorithms, which were used for improving the algorithm performance in the following chapters.

Chapter 3.

Novel PSO-Based Algorithms Using Selective Hybridization

This chapter[2] presents a new approach to improve the performance of PSO-based algorithms in solving an AUV path planning problem by using selective hybridization of DE. The proposed algorithms carry out the DE operation selectively on a number of particles to enhance the swarm's searching ability and resistance to local minima while maintaining a low computational requirement. To analyse the algorithms, an empirical study and a benchmark study based on several non-linear continuous test functions are presented. The algorithms were applied in an offline AUV path planner to generate a time-optimal path that can safely guide the AUV through an environment with *a priori* detected obstacles and time-invariant non-uniform currents. The performances of the algorithms were evaluated and benchmarked by using Monte Carlo methods and Kruskal-Wallis ANOVA tests.

3.1 Selective Hybridization

The hybridization of DE in PSO and QPSO, which were proposed by Zhang and Xie (2003) and Fu et al. (2013), can greatly improve the algorithms' searching ability and resistance to local minima, resulting in better solution quality for path planning problems as shown in Chapter 2. However, these improvements come at the cost of higher runtime required to obtain the solutions. The computational load of the DE-hybridized algorithms increases drastically when the complexity and dimensions of a problem increase. This is caused by the greedy selection operator used in the DE operation, which requires the fitness of all particles to be evaluated twice in every iteration. As the fitness evaluation process usually

[2] This chapter was modified from the following publication: Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., & Bose, N. (2020). Particle swarm optimization algorithms with selective differential evolution for AUV path planning. *International Journal of Robotics and Automation (IJRA)*, 9(2), 94–112.

contributes to the majority of the runtime of a PSO-based algorithm (Sun et al., 2012), the negative impact of the greedy selection operator on the computational load cannot be overlooked. To reduce the negative impact of DE hybridization on PSO-based algorithms, a selective hybridization scheme was proposed to develop the following algorithms:

- SDEPSO (PSO with selective DE hybridization)
- SDEAPSO (PSO with adaptive factor and selective DE hybridization)
- SDEQPSO (QPSO with selective DE hybridization)

Using a selective scheme, these proposed algorithms can apply the DE operation to a number of selected particles only, instead of the entire swarm. The number of particles selected for DE operation, N_s, can be controlled by a selection factor S as shown in Equation (3.1).

$$N_s = N \times S, \quad S \in [0,1] \tag{3.1}$$

In the proposed algorithms, the DE operation was modified by replacing the greedy selection operator with a natural selection operator. After the position update and fitness evaluation of particles, the proposed DE operation can initiate by sorting all the particles in the entire swarm according to their personal best positions. Next, a number of selected particles with the best fitness underwent the mutation and crossover operators to generate the same number of trial vectors. The trial vectors were then subjected to a natural selection operator, in which the same number of particles with the worst fitness were replaced by the trial vectors.

As only the worst particles were replaced in this process, all the potentially best solutions can never deteriorate. Furthermore, the computational cost of the algorithms can be maintained at a reasonable level because the natural selection operator did not involve fitness comparison between the particles, which requires an additional particle fitness evaluation step in every iteration. The DE operation with natural selection can increase the diversity and the evolutionary rate of the entire swarm by eliminating the least desirable solutions, hence leading to a faster and better global convergence theoretically.

The selective DE hybridization was applied to PSO and QPSO algorithms to develop the SDEPSO and SDEQPSO algorithms in this paper. Inspired by the APSO algorithm proposed by Zhan et al. (2009), another algorithm, namely SDEAPSO, was developed by adding an adaptive mechanism to control the inertia weight and acceleration coefficients in PSO. The application of SDEPSO, SDEAPSO and SDEQPSO algorithms in AUV path planning can be conducted as described in the following pseudocode.

Step 1. **Define** the algorithm parameters and ocean environments.

Step 2. **Initialise** a group of candidate paths by generating particles with random positions in Equation (2.1). Set *pbest* to the current particle positions.

Step 3. **While** the stop criteria are not met,

Step 3.1 **For** $t = 1, 2, \ldots, t_{max}$,

SDEPSO	*SDEAPSO*	*SDEQPSO*
Evaluate particle fitness $F(X_i^t)$.	Evaluate particle fitness $F(X_i^t)$.	Compute *mbest* using Equation (2.10).
Update *pbest* and *gbest* using Equations (2.4) and (2.5) respectively.	Update *pbest* and *gbest* using Equations (2.4) and (2.5) respectively.	Evaluate particle fitness $F(X_i^t)$.
Update w using Equation (2.6).	Update w, C_1 and C_2 using Equations (2.15) and (2.16) respectively.	Update *pbest* and *gbest* using Equations (2.4) and (2.5) respectively.
		Update β using Equation (2.12).

Step 3.2 **For** each particle $i = 1, 2, \ldots, N$,

SDEPSO	*SDEAPSO*	*SDEQPSO*
Update particle velocity and position using Equation (2.2) and (2.3) respectively.	Update particle velocity and position using Equation (2.2) and (2.3) respectively.	Update particle position using Equation (2.11).

End for

Step 3.3 **Sort** all particles according to the fitness of their personal best positions.

Step 3.4 **For** $k = 1, 2, \ldots, N_s^{th}$ best performing particle,

Mutation: Generate mutated vector U_k^t using Equation (2.25).

Crossover: Generate trial vector T_k^t using Equation (2.26).

Natural selection: Replace k^{th} worst-performing particle with T_k^t.

End for

End for

End while

Step 4. **Output** *gbest* that holds the optimal path when the stop criteria are met.

3.2 Complexity Analysis

The time complexity of the proposed algorithms can be measured by counting the number of primitive operations in the algorithm. By referring to the pseudocode of the proposed algorithms, the number of operations can be counted as follows:

- Initialisation in Step 2 contributes to N operations.
- Step 3.1 consists of several operations. Fitness evaluation contributes to N operations. Finding *pbest* requires N log(N) operations. Finding *gbest* requires log(N) operations. Updating the equation coefficients contributes to one operation. SDEQPSO requires N additional operations to calculate *mbest*.
- In Step 3.2, SDEPSO and SDEAPSO perform N loops with 14 operations; SDEQPSO performs N loops with 12 operations.
- Step 3.3 contributes to log(N) operations.
- Step 3.4 performs N_S loops with 8 operations.

Steps 1 – 3.2 are the standard operations in the basic PSO, APSO and QPSO algorithms, whereas Step 3.3 and 3.4 are the additional operations introduced by the selective DE operation. Big O notation was used in this work to analyse the asymptotic upper bound of time complexity, which can indicate the computational time of the algorithm in the worst-case scenario. When finding the O notation, the lower order terms in the number of operations are negligible because their impacts on computational time are relatively insignificant for large N (Raphael & Smith, 2003). The highest-order term in the operation is N log(N) in Step 3.1, and it is performed $t_{\max}$ times to check the stop criteria. The operations added by the selective DE operation (Step 3.3 and 3.4) are of lower order and do not have a significant impact on the time complexity. Thus, the complexity of the proposed algorithms in linear form is $O(N \log(N) \cdot t_{\max})$, similar to the standard PSO algorithm. PSO-based algorithms have two inner loops when going through the population of N particles, and one outer loop for $t_{\max}$ iterations; this renders the time complexity to be $O(N^2 \cdot t_{\max})$ in the extreme case. The spatial complexity of the algorithms is $O(N^2)$, which depends on the population size.

3.3 Benchmark Functions

Metaheuristic algorithms such as PSO-based algorithms can be evaluated empirically by comparing their performances in solving the optimization problem of a test function. The development and evaluation of an algorithm for a specific problem should be based on test functions of similar classes and properties because it is unlikely for an algorithm to perform consistently for every class of optimization problem according to the "no free lunch" (NFL) theorem (Wolpert & Macready, 1997). In addition to the AUV path planning problem, non-linear continuous test functions were used to study and benchmark the characteristics of the proposed algorithms. These benchmark functions were selected based on their resemblances to the properties of the path planning problem, which has the following properties:

- Multimodal with deceptive local minima and one global minimum: a path planning problem usually consists of multiple suboptimal paths and a Pareto-optimal path.
- Multi-dimensional: the dimension of a path planning problem is dependent on the number of control waypoints along the path.

Four test functions that exhibit the abovementioned properties were chosen for benchmarking in this study. These test functions, which are minimization problems, are commonly used to evaluate the characteristics of optimization algorithms. Table 3.1 provides information about the selected benchmark functions. The dimensions of all the functions were set to 20.

Table 3.1: Benchmark functions.

Notation	Name	Function formulation	Boundary interval	Global minimum
$F1$	Griewank (Locatelli, 2003)	$F(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]	$F(x) = 0$, at $x = (0, \ldots,0)$
$F2$	Rastrigin (Mühlenbein et al., 1991)	$F(x)=10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)]$	[-5.12, 5.12]	$F(x) = 0$, at $x = (0, \ldots,0)$

Table 3.1 (continued).

$F3$	Ackley (Ackley, 1987)	$F(x) = -20e^{\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)} - e^{\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right)} + 20 + e$	[-32, 32]	$F(x) = 0$, at $x = (0, \ldots, 0)$
$F4$	Schwefel (Bäck & Schwefel, 1993)	$F(x) = 418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]	$F(x) = 0$, at $x = (420.9687, \ldots, 420.9687)$

3.4 Empirical Study on Parameter Selection

In the selectively DE-hybridized algorithms, the number of best-performing particles that undergo the DE operation and the number of worst-performing particles that are replaced during natural selection can be determined by the selection factor *S*. Therefore, *S* can be fine-tuned to control the diversity of the population. To study the effects of *S* on the algorithm performance, an empirical study was conducted on SDEPSO by using a range of *S*, which can be a positive number that is less than 1.0.

When $S = 0$, DE hybridization is prohibited in the algorithm. When $S = 1$, the DE operation is conducted on the entire swarm, and the entire swarm is replaced during natural selection. This implies that all solutions generated from the PSO operation will be discarded, which is undesirable. Therefore, the empirical study included *S* values ranging from 0 to 0.9 to investigate the algorithm performance when 0% – 90% of the particles undergo the DE operation. The performance for $S = 0$ was studied for comparison purposes. Through a 1000-run Monte Carlo simulation with 100 (max) iterations and a swarm size of 150 particles, the performance of SDEPSO under different *S* settings was evaluated by solving the optimization problems of the benchmark functions and the path planning problem in 2D and 3D scenarios. The path planning problem followed the formulation described in Chapter 2.

Prior to evaluating the algorithm performance, the Shapiro-Wilk test with a significance level of 0.05 was performed to examine the normality of the obtained simulation results. The normality test revealed that the data was not normally distributed. Therefore, medians

can be used as an indicator of solution quality. Table 3.2 shows the medians of the obtained fitness (*Med*) and the best-known fitness (*Best*) obtained for all the problems under different *S*. Minimization was required in all the problems and thus a lower fitness value indicates a higher solution quality and hence a stronger searching ability. The best-performing results for each problem are in bold.

Table 3.2: Empirical study results.

Selection factor, *S*	***F*1**		***F*2**		***F*3**		***F*4**		**2D path planning** ($\times 10^2$)		**3D Path planning** ($\times 10^2$)	
	Med	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*
0	0.86	0.25	1.28	0.41	0.19	0.06	2.61	1.54	3.07	2.97	3.36	3.30
0.10	0.58	**0.13**	1.22	0.42	**0.15**	0.06	2.22	1.20	3.06	2.99	3.20	3.14
0.20	**0.56**	**0.13**	1.20	0.50	**0.15**	0.07	2.08	0.69	3.01	2.97	3.34	**3.13**
0.30	0.63	0.19	**1.15**	**0.21**	0.17	**0.05**	1.89	0.85	**2.98**	**2.91**	**3.18**	3.14
0.40	0.68	0.34	1.29	0.51	0.23	0.08	1.90	0.81	3.06	2.96	3.30	3.15
0.50	0.66	0.30	1.27	0.40	0.26	0.12	1.71	0.65	3.12	3.02	3.44	3.15
0.60	0.73	0.14	1.41	0.52	0.32	0.11	1.70	0.63	3.05	2.98	3.42	3.18
0.70	0.80	0.34	1.61	0.50	0.47	0.10	1.71	0.60	3.00	2.98	3.33	3.19
0.80	0.87	0.43	1.59	0.84	0.74	0.26	1.51	0.57	3.05	2.97	3.22	3.19
0.90	1.00	0.85	1.77	0.61	1.67	0.43	**1.27**	**0.48**	3.08	2.97	3.35	3.25

It was found that the computational time remained relatively consistent when *S* changed. However, Table 3.2 shows that the performance of the algorithms varied greatly as *S* increases, and the variations were not consistent for all the problems. For the majority of the problems, the best results were identified to be in the range of $S = 0.1 – 0.3$, except for *F*4. This can be explained by the geometry of the Schwefel function *F*4, which has all of its local minima and the global minimum spread far apart from one another. Effective optimization of this function requires an algorithm that promotes larger solution diversity (higher *S*) so that it provides a jumping-out ability to prevent trapping in deceptive local minima.

The simulation results complied with the NFL theorem, which suggests that no single algorithm can generate better performance than any other algorithms for every problem. The improved algorithm performance in one class of problem is not necessarily consistent in all kinds of problems; instead, it is exactly traded with performance in another class of problem (Wolpert & Macready, 1997). Although all the test functions selected for benchmarking have similar properties (multimodal and multi-dimensional), the geometries of their problem spaces are different. Therefore, the setting of S should be adjusted accordingly for different optimization problems.

Based on this empirical study, it can be deduced that the optimal setting of S for the majority of the tested problems is in the range of 0.1 – 0.3. More specifically for the path planning problem, the setting of $S = 0.3$ was found to be appropriate and effective.

3.5 Benchmark Study

The benchmark functions were used to evaluate the proposed algorithms in this study. Through a 1000-run Monte Carlo simulation with 100 (max) iterations and a swarm size of 150 particles, the performances of the proposed algorithms in solving the optimization problems of the four benchmark functions were compared with other existing PSO-based algorithms. At each run, the initial particle positions for all the problems were randomly generated based on uniform distribution within the boundary intervals given in Table 3.3.

As the data was not normally distributed according to the Shapiro-Wilk test, the Kruskal-Wallis test (McKight & Najab, 2010), which is a non-parametric ANOVA, was used with a significance level of 0.05 to rank the algorithm performance based on the obtained fitness values. The ranking procedure used the Holm–Bonferroni 'stepdown' approach (Hochberg & Tamhane, 1987), which is best suited for all pairwise comparisons when the confidence intervals are not needed and sample sizes are equal (Sun et al., 2012). The algorithms were given the same rank if they were not statistically different from one another. The medians of the obtained fitness (*Med*), the ANOVA ranks (R) and the medians of runtime required (T) were tabulated in Table 3.3. The medians of the top two best-performing results for each problem are in bold. The overall performance of an algorithm can be given by its total ranks ($\#R$), which was calculated from the summation of the ranks of the algorithm for all the problems.

Table 3.3: Benchmark study results.

Algorithm	*F1*			*F2*			*F3*			*F4*			#*R*
	Med	*R*	*T(s)*	*Med*	*R*	*T(s)*	*Med*	*R*	*T(s)*	*Med*	*R*	*T(s)*	
PSO	0.658	8	0.10	1.372	5	0.12	0.453	8	0.10	3.617	5	0.13	26
QPSO	0.089	3	0.16	1.791	6	0.15	0.005	1	0.17	4.555	8	0.19	18
APSO	0.100	4	0.16	1.219	4	0.16	0.041	5	0.18	3.606	5	0.20	18
DEPSO	0.634	6	0.43	**1.140**	1	0.55	0.166	6	0.42	**1.781**	1	0.47	14
DEQPSO	**0.064**	1	0.51	2.092	7	0.50	**0.002**	1	0.49	3.023	4	0.56	13
SDEPSO	0.629	6	0.11	**1.149**	1	0.14	0.172	6	0.18	**1.891**	2	0.20	15
SDEAPSO	0.098	4	0.16	1.196	3	0.16	0.035	4	0.18	2.031	3	0.27	14
SDEQPSO	**0.072**	2	0.18	2.125	7	0.18	**0.002**	1	0.19	3.594	5	0.27	15

Based on Table 3.3, there was no single algorithm that can achieve the best results for all the problems. This observation agrees with the NFL theory. For the Griewank function (F_1), the DEQPSO algorithm generated the best solution. Satisfactory solutions of F_1 were also produced by APSO, SDEAPSO, QPSO, DEQPSO, and SDEQPSO. This indicates that having an adaptive mechanism in PSO or the quantum behaviour of QPSO was beneficial for solving this problem. The DEPSO and SDEPSO algorithms generated equally good solution qualities for the Rastrigin function (F_2). For the Ackley function (F_3), the QPSO-based algorithms, *i.e.*, QPSO, DEQPSO and SDEQPSO, produced the best solution qualities; this was followed closely by the APSO-based algorithms. As far as the Schwefel function (F_4) was concerned, only DEPSO, SDEPSO and SDEAPSO can generate satisfactory results, while all the other algorithms showed inferior performances.

The total ranks of the algorithms revealed that DEQPSO provided a better overall solution quality than other algorithms. The performances of DEPSO and SDEAPSO were ranked second, followed by SDEPSO and SDEQPSO. More importantly, the simulation results showed that the fully DE-hybridized algorithms required significantly higher runtime to obtain the solutions. The selectively DE-hybridized algorithms can produce good solution qualities while maintaining a reasonably similar computational requirement as the PSO, QPSO and APSO algorithms.

3.6 Numerical Simulations

Numerical simulations were performed to benchmark the performance of the proposed algorithms for the AUV path planning problem, which was formulated based on the description in Section 2.2. In addition to PSO-based algorithms, the proposed algorithms were also benchmarked against another path planning technique, RRT*, and other metaheuristic algorithms, including DE, Ant Colony Optimization (ACO) (Mirjalili et al., 2020), Firefly Algorithm (FA) (MahmoudZadeh et al., 2018), and Genetic Algorithm (GA) (Alvarez et al., 2004).

The benchmark study employed the Monte Carlo method with 1000 simulation runs. In each simulation run, the algorithms were allowed to iterate up to 100 times with a predefined stopping threshold. The population size of all the algorithms was set to 150 particles. The simulations were conducted in 2D domains followed by 3D domains. The problem spaces of the simulations were assumed to be an ocean field that is 500×500 square metres for 2D and 500×500×500 cubic metres for 3D. Non-uniform ocean currents and static obstacles of different sizes were present in the problem space. The current field was generated based on the data obtained from the field experiment conducted at Beauty Point, Tasmania, Australia. The AUV was required to travel from a starting point to a target with a preset water-referenced velocity of 1.5 m/s. The simulations were conducted on a machine with Intel Core i5-6300U CPU @ 2.4GHz with 8GB RAM.

3.6.1 Benchmark of Path Planners

The performances of the path planners were benchmarked based on their solution qualities, stabilities, and computational requirements. These properties were reflected by the fitness values of the solutions obtained and the runtime required to obtain the solutions. A lower fitness value is desirable because it represents the time required by the AUV to arrive at the target by following the generated path.

The simulation results were not normally distributed based on the Shapiro-Wilk normality test with a significance level of 0.05. Therefore, non-parametric boxplots were used for analysis. The boxplots in Figure 3.1 depict the distribution of fitness values obtained by all the algorithms in 2D and 3D scenarios. The data ranges (black whiskers) and percentile ranges (blue boxes) of the boxplots indicate the stabilities of the algorithms, while the medians (red lines inside the boxes) show the solution qualities and searching abilities. The

individual best fitness obtained by each algorithm in the 1000-run simulations can be identified by the extreme lowest end of each whisker. The best-known (lowest) fitness value among all the algorithms in the simulations is represented by the green cross sign in the boxplots.

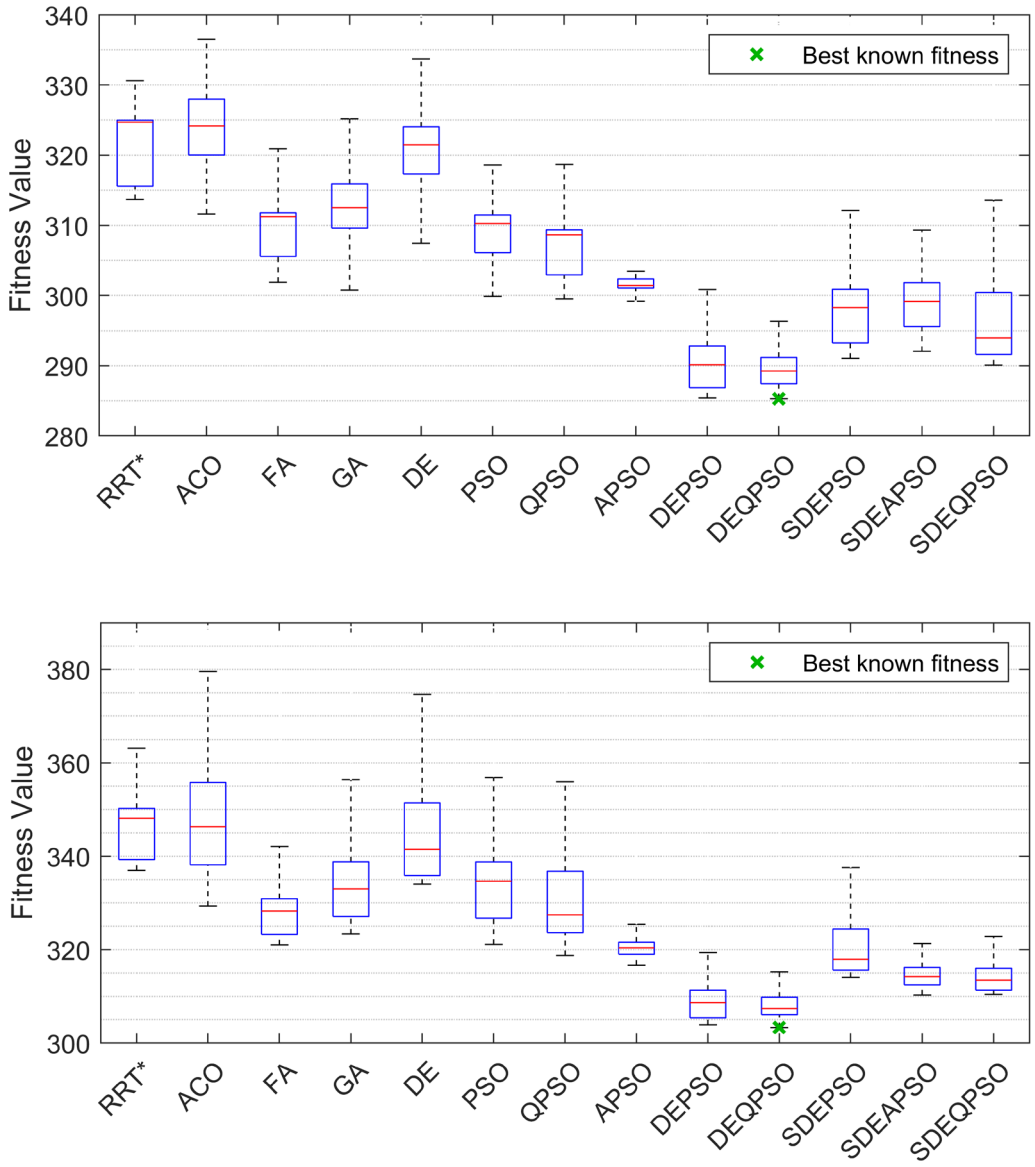


Figure 3.1: Fitness values obtained in 2D (top) and 3D (bottom) scenarios.

The Kruskal-Wallis ANOVA procedure with a significance level of 0.05 was used to rank the algorithms based on their obtained fitness values. The ranking was based on the Holm–Bonferroni step-down method. The algorithms were given the same rank if they were not statistically different from one another. Table 3.4 shows the details of the simulation results, including the median of the obtained fitness (*Med*), the best-known fitness (*Best*), the

interquartile range (*IQR*), the ANOVA rank (*R*), the median of runtime (*T*) and the total rank (#*R*). The total ranks were calculated from the summation of the ranks in the 2D and 3D scenarios. The ranking of the algorithms did not consider the impact of runtime.

Based on Figure 3.1 and Table 3.4, most of the PSO-based algorithms showed better solution quality than RRT* and other metaheuristic algorithms, with the exception of the standard PSO being outperformed by FA. Despite having lower solution quality, RRT* required the shortest runtime in both 2D and 3D scenarios. It can be observed that all the PSO and QPSO variants produced better solution qualities than the standard PSO and QPSO. DEPSO and DEQPSO outperformed all the other algorithms by achieving the lowest medians for fitness value in both 2D and 3D. The total ranks of DEPSO and DEQPSO suggest that the two fully DE-hybridized algorithms were able to produce the best solution qualities for the path planning problem. However, the runtime of DEPSO and DEQPSO was significantly higher than all the other algorithms due to the high computational requirements of the greedy selection operator.

Table 3.4: Simulation results for the benchmark of path planners.

Algorithm	**2D**					**3D**					**#*R***
	Med ($\times 10^2$)	*Best* ($\times 10^2$)	*IQR*	*R*	*T(s)*	*Med* ($\times 10^2$)	*Best* ($\times 10^2$)	*IQR*	*R*	*T(s)*	
RRT*	3.25	3.14	9.4	11	4.8	3.48	3.37	10.9	11	14.3	22
ACO	3.24	3.12	8.0	13	9.4	3.46	3.29	17.6	11	41.3	24
FA	3.11	3.02	6.2	8	9.2	3.28	3.21	7.7	7	41.2	15
GA	3.13	2.98	6.3	10	12.3	3.33	3.23	11.7	9	48.3	19
DE	3.21	3.05	6.7	11	12.8	3.41	3.34	15.5	11	53.6	22
PSO	3.10	3.00	5.4	8	10.7	3.35	3.21	12.1	9	34.6	17
QPSO	3.09	3.00	6.4	7	9.9	3.27	3.19	13.2	7	30.9	14
APSO	3.01	2.92	1.3	5	10.8	3.20	3.17	2.6	5	37.7	10
DEPSO	2.90	2.85	5.9	1	22.4	3.09	3.04	5.9	1	69.0	2
DEQPSO	2.89	2.85	3.7	1	20.8	3.07	3.03	3.7	1	76.7	2
SDEPSO	2.98	2.91	7.7	6	12.8	3.18	3.14	8.8	5	35.7	11
SDEAPSO	2.99	2.92	6.2	3	14.9	3.14	3.10	3.7	4	38.8	7
SDEQPSO	2.94	2.90	7.8	3	13.7	3.13	3.10	4.7	3	37.6	6

The solution qualities of SDEAPSO and SDEQPSO were second to the fully DE-hybridized algorithms; they were ranked similarly in 2D based on the ANOVA ranking. APSO provided a better solution quality than SDEPSO in 2D. It is worth noting that APSO achieved the lowest interquartile range in both 2D and 3D, indicating the highest stability among all the algorithms. In the 3D scenario, SDEQPSO was ranked slightly higher than SDEAPSO, while SDEPSO was ranked similar to APSO.

SDEQPSO and SDEAPSO were able to offer up to a 10% improvement in the fitness values of solutions when compared to other tested algorithms. The total ranks of the overall performance in both 2D and 3D revealed that the solution qualities of SDEQPSO and SDEAPSO were ranked as the third and the fourth, respectively. More importantly, the runtime of the two selectively DE-hybridized algorithms were very close to other PSO-based algorithms and were significantly (up to 50%) lower than the fully DE-hybridized algorithms. These indicate the higher computational efficiency of SDEQPSO and SDEAPSO in solving the path planning problem because they were able to produce solution quality that was very close to DEPSO and DEQPSO, while having a significantly lower computational requirement. In terms of problem size, the runtime required by the path planner was considered short, particularly in comparison to the computational time required for sensing and estimating the vehicle's operational environments.

3.7 Vehicle Path Verification

For verification purposes, the path solutions generated by the proposed AUV path planner were used to produce a reference trajectory for the dynamic model of the Hydroid REMUS 100, a 1.7-metre-long torpedo-shaped AUV. The detailed specifications of the REMUS 100 AUV can be found in Appendix A. This section outlines the dynamic model and the path following controller used.

3.7.1 Dynamic Model

Based on the vectorial representation described by Fossen (1999) and the standard formulation of SNAME (Society of Naval Architects and Marine Engineers), the 6 DOF equation of motion for a typical AUV can be modelled using Equations (3.2) and (3.3).

$$\dot{\eta} = \begin{bmatrix} R(\eta_2) & 0_{3\times3} \\ 0_{3\times3} & T(\eta_2) \end{bmatrix} \upsilon \tag{3.2}$$

$$M\dot{\upsilon}_r + C(\upsilon_r)\upsilon_r + D(\upsilon_r)\upsilon_r + g(\eta) = \tau \tag{3.3}$$

$$\eta = [\eta_1 \quad \eta_2]^{\mathrm{T}} = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^{\mathrm{T}} \tag{3.4}$$

$$\upsilon = [\upsilon_1 \quad \upsilon_2]^{\mathrm{T}} = \left[\dot{x}_b \quad \dot{y}_b \quad \dot{z}_b \quad \dot{\phi}_b \quad \dot{\theta}_b \quad \dot{\psi}_b\right]^{\mathrm{T}} \tag{3.5}$$

$$\upsilon_r = \upsilon - V_c \tag{3.6}$$

where $R(\eta_2)$ and $T(\eta_2)$ are the rotation matrices between inertial and body-fixed reference frames for translational velocities and angular velocities, respectively. η includes the position η_1 and the orientation η_2 of the vehicle with respect to the inertial reference frame as described in Equation (3.4), while the derivative of η in Equation (3.2) represents their rates of change. υ includes the translational velocities υ_1 and the rotational velocities υ_2 of the vehicle with respect to the body-fixed reference frame as described in Equation (3.5). υ_r is the relative velocity vector derived from the velocity vector of ocean currents V_c as defined in Equation (3.6).

Equation (3.3) defines the kinetics of an AUV, in which M and $C(\upsilon)$ denote the inertial and Coriolis matrices (both including rigid body and added mass) respectively. $D(\upsilon)$ is the hydrodynamics damping matrix, and $g(\eta)$ is the hydrostatics restoring forces. The actuators' control forces are represented by τ. The REMUS 100 AUV has a pair of horizontal stern planes for pitch control and a pair of vertical rudder planes for heading control. Using the derivation of Prestero (2001), the control forces and moments of the REMUS 100 AUV can be given as follows:

$$\tau = \left[X_{prop} \quad Y_{rudder} \quad Z_{stern} \quad K_{prop} \quad M_{stern} \quad N_{rudder}\right]^{\mathrm{T}} \tag{3.7}$$

$$X_{prop} = K_T \rho_w D_{prop}{}^4 n_{rps}{}^2 \tag{3.8}$$

$$Y_{rudder} = 0.5 \rho_w c_{L,\alpha} S_{fin} \left[\dot{x}_b^2 \delta_{rudder} - \dot{x}_b \dot{y}_b - x_{fin} (\dot{x}_b \dot{\psi}_b)\right] \tag{3.9}$$

$$Z_{stern} = 0.5 \rho_w c_{L,\alpha} S_{fin} \left[\dot{x}_b^2 \delta_{stern} - \dot{x}_b \dot{z}_b - x_{fin} \left(\dot{x}_b \dot{\theta}_b\right)\right] \tag{3.10}$$

$$K_{prop} = K_Q \rho_w D_{prop}{}^5 n_{rps}{}^2 \tag{3.11}$$

$$M_{stern} = 0.5 \rho_w c_{L,\alpha} S_{fin} x_{fin} \left[\dot{x}_b^2 \delta_{stern} - \dot{x}_b \dot{z}_b - x_{fin} \left(\dot{x}_b \dot{\theta}_b\right)\right] \tag{3.12}$$

$$N_{rudder} = 0.5 \rho_w c_{L,\alpha} S_{fin} x_{fin} \left[\dot{x}_b^2 \delta_{rudder} - \dot{x}_b \dot{y}_b - x_{fin} (\dot{x}_b \dot{\psi}_b)\right] \tag{3.13}$$

where ρ_w is the density of water, K_T and K_Q are the thrust and torque coefficients of the propeller, D is the propeller diameter, and n_{rps} is the propeller speed in revolutions per second. The dynamic model used the K_T and K_Q coefficients obtained for the REMUS 100 AUV by Allen et al. (2000). For the forces and moments of the control fins, S_{fin} is the fin planform area, x_{fin} is the body-referenced x-coordinate of the fin, and $c_{L,\alpha}$ is the lift coefficient at the angle of attack α. δ_{rudder} and δ_{stern} are the body-referenced angles of the rudder and stern planes relative to the vehicle x-axis, respectively. The coefficients calculated by Prestero (2001) were used in the dynamic model.

3.7.2 Path Following Guidance and Controller

The path following controller of the AUV model used the integral line-of-sight (iLOS) guidance law to set the yaw and pitch angles for following the planned path. The controller enabled the AUV to shape its convergence towards the planned path in the presence of ocean currents and environmental disturbance by using the iLOS guidance law (Caharija et al., 2012). The desired iLOS yaw angle (heading) ψ_d and pitch angle θ_d can be given by Equations (3.14) and (3.15).

$$\psi_d(e) \triangleq \arctan\left(\frac{e + K_{i,y} e_{\text{int}}}{\Delta_y}\right), \qquad K_{i,y},\ \Delta_y > 0$$
$$\dot{e}_{\text{int}} = \frac{\Delta_y e}{\left(e + K_{i,y} e_{\text{int}}\right)^2 + \Delta_y^2} \tag{3.14}$$

$$\theta_d(h) \triangleq \arctan\left(\frac{h + K_{i,z} h_{\text{int}}}{\Delta_z}\right), \qquad K_{i,z},\ \Delta_z > 0$$
$$\dot{h}_{\text{int}} = \frac{\Delta_z h}{\left(h + K_{i,z} h_{\text{int}}\right)^2 + \Delta_z^2} \tag{3.15}$$

where e is the cross-track error, and h is the vertical-track error. $K_{i,y}$ and $K_{i,z}$ are the integral gains, while Δ_y and Δ_z represent the look-ahead distances for iLOS heading and pitch, respectively. The integral terms of cross-track error e_{int} and vertical-track error h_{int} produce non-zero ψ_d and θ_d even when the AUV is on the planned path, allowing the vehicle to counteract any effects of ocean current with the necessary side-slip and pitch angles. The rates of integral terms $\dot{e}_{int}$ and $\dot{h}_{int}$ reduce the integral action with large cross-track and vertical-track errors (*i.e.*, the vehicle is far from the planned path) to minimize the risk of integrator wind-up.

To follow the desired iLOS heading and pitch, the AUV model used a control system with the speed, heading and pitch controllers described by Caharija et al. (2012). The speed controller used a feedback linearizing proportional (P) controller to control the propeller shaft speed. The heading and pitch controllers were two feedback linearizing proportional-derivative (PD) controllers that control the rudder and stern planes, respectively. In order to prevent excessive commands to the controllers, third-order low-pass filters were implemented as reference models before passing the desired setpoint to the controllers.

3.7.3 Verification Results

The feasibility of the path solutions was first checked against the vehicular constraints of REMUS 100, which has a minimum turning radius of 8.1 metres in the worst-case scenario (Eng et al., 2016). The curvature radius of a feasible path must be higher than the minimum turning radius. Figure 3.2 shows the paths generated by the SDEQPSO algorithm satisfied the AUV's constraints.

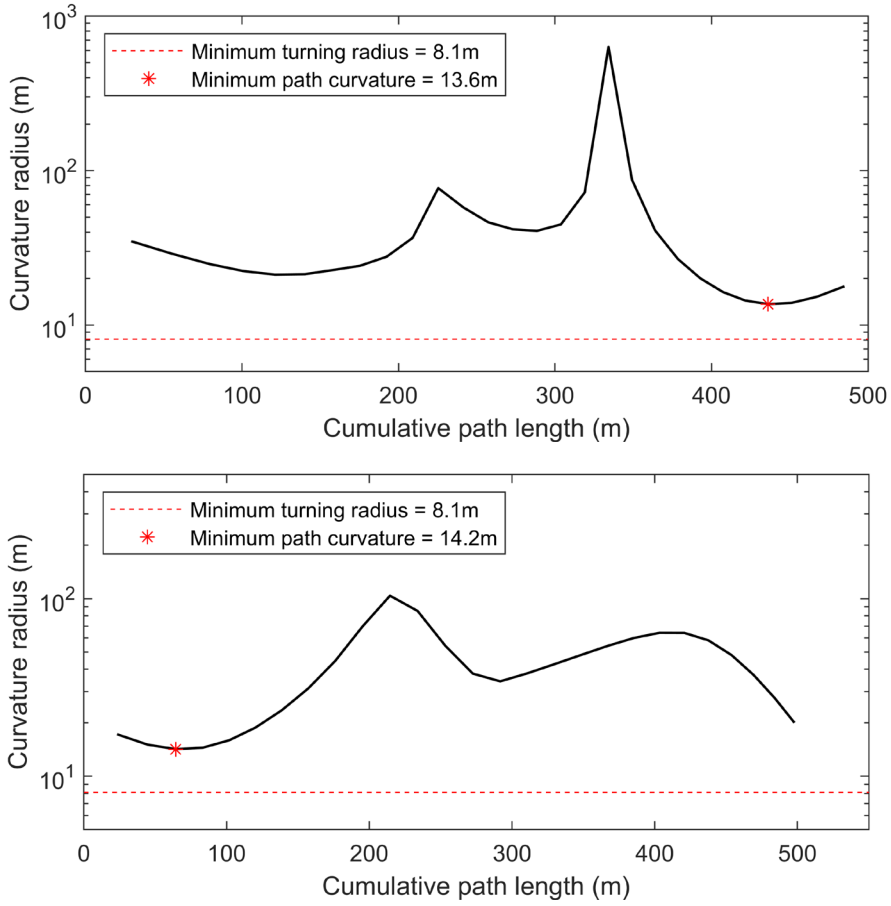


Figure 3.2: Curvature radius of planned paths for 2D (top) and 3D (bottom).

Next, the 2D and 3D solutions generated by SDEQPSO were analysed by comparison with the simulated paths in Figure 3.3. The AUV was required to travel from the starting point (green square) to the target (pink star) without running into obstacles while trying to exploit favourable ocean currents to assist the AUV motion by following time-optimal paths. Using a colour bar in the 2D result of Figure 3.3, the blue regions represent favourable currents, while the red regions denote less favourable currents. The favourable currents were defined as positive and flowing towards the top right corner, hence pushing the vehicle from its starting position towards the target. The less favourable currents were defined as negative and flowing in the opposite direction. In both results, the solid sections of the planned paths indicate that the favourable currents have a positive effect on the AUV motions while the dashed sections suggest otherwise. Figure 3.3 shows that the AUV was able to follow time-optimal paths to exploit the favourable currents and avoid the less favourable currents. The simulated paths closely resembled the planned paths in both scenarios.

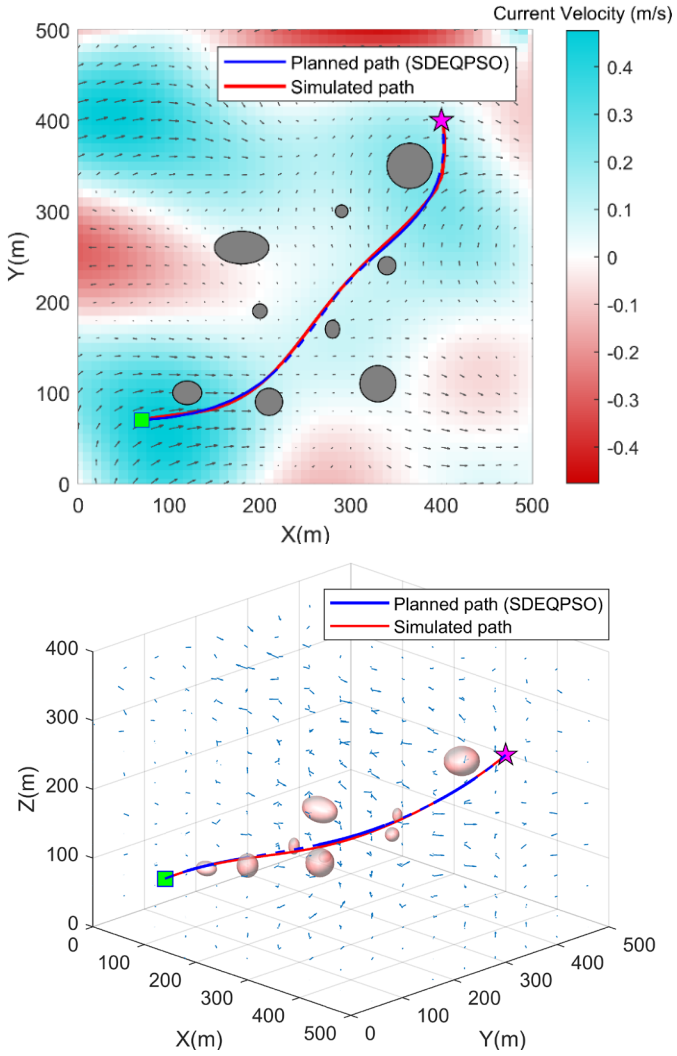


Figure 3.3: Verification of path solutions in 2D (top) and 3D (bottom).

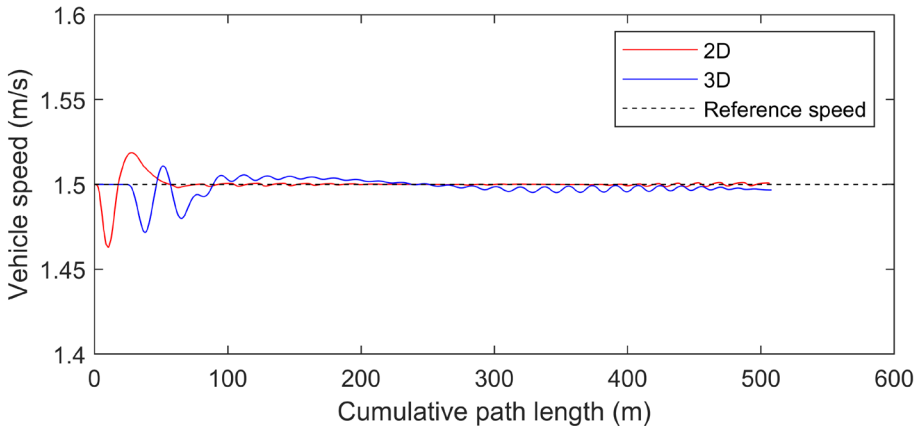


Figure 3.4: Vehicle speed of REMUS 100 in 2D and 3D scenarios

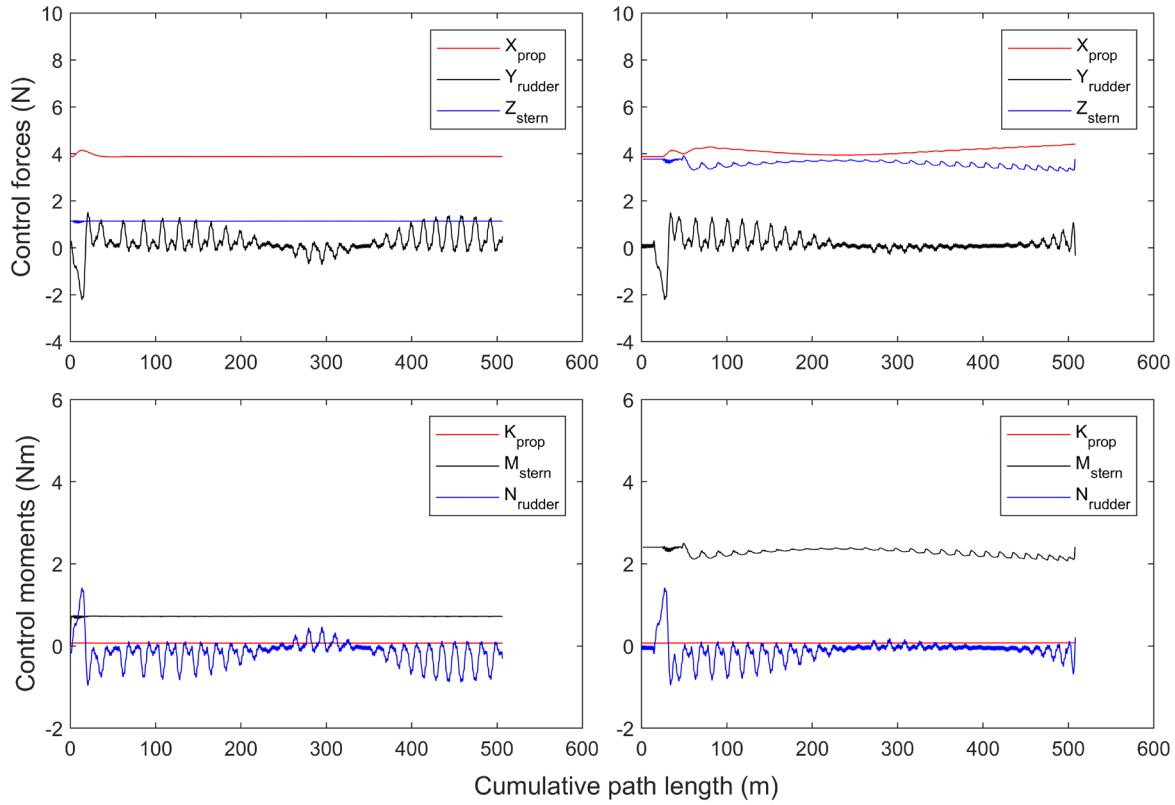


Figure 3.5: Control forces and moments of REMUS 100 in 2D (top-left and bottom-left) and 3D (top-right and bottom-right).

Figure 3.4 shows the variation of vehicle speed when following the planned paths in the 2D and 3D scenarios, while Figure 3.5 depicts the control forces and moments required for following the paths. The AUV maintained an average speed of 1.5 m/s in both scenarios. Next, the total track errors of the simulated paths relative to the planned paths for the 2D and 3D scenarios are graphed in Figure 3.6. The total track error in 2D is simply cross-track error, whereas the total track error in 3D is the total resultant error calculated from cross-

track and vertical-track errors. The errors for both scenarios were well below 1 metre, proving that the AUV was able to follow the planned paths closely. Hence, the simulation results show that the path solutions generated by the proposed algorithm were smooth and dynamically feasible for path planning applications.

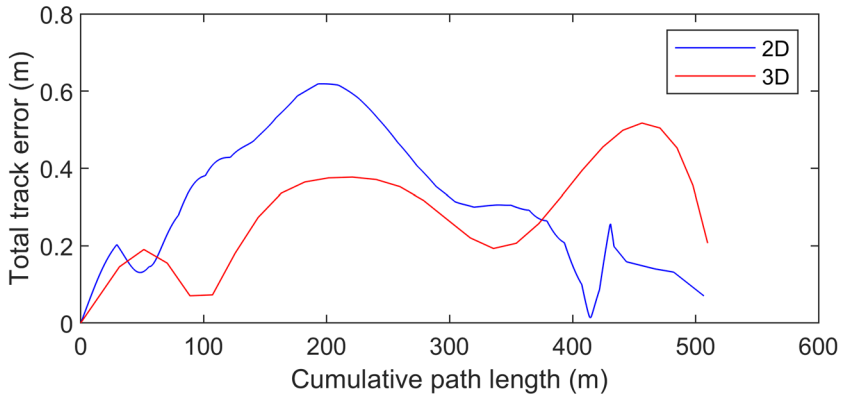


Figure 3.6: Total track error of simulated paths relative to planned paths.

3.8 Chapter Summary

By selectively hybridizing with DE, this chapter presents new variants of the PSO algorithm with an improved searching ability for the global minimum path of an AUV without increasing the computational requirements. The proposed algorithms were benchmarked against other algorithms in offline AUV path planning scenarios, which consist of *a priori* known obstacles of different sizes and non-uniform ocean currents. Based on the Monte Carlo simulations and ANOVA procedures, the SDEAPSO and SDEQPSO algorithms were able to surpass most of the tested algorithms (up to a 10% improvement in solution quality). SDEQPSO and SDEAPSO achieved similar performance to DEPSO and DEQPSO algorithms in terms of solution quality and stability, while having a significantly lower computational requirement (up to a 50% reduction in algorithm runtime). Most importantly, the simulation results showed that the generated time-optimal paths were smooth, feasible and able to exploit ocean currents to improve the vehicle performance.

The selectively DE-hybridized PSO algorithms proposed in this study are efficient for solving non-deterministic polynomial-time (NP)-hard problems, such as the path planning problem. The simulation assumed *a priori* known environments to represent the minimum capability of a path planner. The algorithms were adapted to solve path replanning problems under more realistic operational conditions in the following chapters.

This page is intentionally left blank.

Chapter 4.

Constrained Path Planning Using Polar Coordinates

This chapter[3] presents the formulation of an objective function for AUV path planning as a constrained optimization problem to improve the computational efficiency of the algorithms proposed in the previous chapter. The search domain of AUV paths was modelled using the polar coordinate system and a combination of hard and soft constraints. This approach allowed the vehicular constraints to be satisfied and facilitated the placement of path nodes to enhance the search efficiency of the path planner. The effect of different types of constraints on the algorithm performance was thoroughly analysed using Monte Carlo simulations and Kruskal-Wallis tests. By combining hard and soft constraints, the path planner effectively generated a time-optimal path that can safely guide an AUV through *a priori* known ocean environments.

4.1 Constraint Handling in Path Planners

In the previous chapters, the objective function of the path planners was based on an unconstrained search domain with penalties applied to ensure obstacle avoidance and vehicular constraints. The path nodes can be freely placed in the problem space as long as they are collision-free and satisfy the vehicular constraints. By combining with an efficient optimization algorithm, the unconstrained objective function can produce a good overall performance for offline planning. However, the search efficiency of a metaheuristic optimization-based path planner also relies on its objective function. A path planner with an inefficient objective function may suffer from the inconsistency and incompleteness of

[3] This chapter was modified from the following publication: Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., Bose, N., & Kim, E. (2019). Constrained path planning of autonomous underwater vehicle using selectively hybridized particle swarm optimization algorithms. *IFAC-PapersOnLine*, 52(21), 315–322.

solutions if the allowable time for planning is limited, which is often the case in real AUV operations. Thus, developing an efficient objective function for path planning is of critical importance for an AUV operating in a highly dynamic and time-constraint scenario, where online path planning is required.

To improve the search efficiency of an optimization-based path planner, its objective function can be constrained to discretise the problem space into smaller search domains. Existing studies proposed various approaches for the discretisation of problem space. A traditional approach, which is compatible with the Cartesian coordinate system, uses uniformly discretised grids to represent the problem space. The grid-based approach can allow a path node to be located on a grid's vertices (Alvarez et al., 2004), centre (Tanakitkorn et al., 2014), edges (Gong et al., 2011; Yao et al., 2018) or anywhere within the grid (Zhang et al., 2014). The grid-based approach is suitable for operational environments that can be simplified into 2D.

Another approach, which requires the use of a polar/spherical coordinate system, uses concentric circles (in 2D) or concentric spheres (in 3D) to discretise the problem space. In this approach, a path node may be placed on the circumference (Hao et al., 2007; Zu et al., 2008) or the annular space between circles (Zeng et al., 2015). Polar and spherical coordinate systems can be more relevant for the path planning of an underactuated torpedo-shaped AUV, which has limited actuation, if not none, for heave and sway motions. The search domain of a path planner that uses polar/spherical coordinate systems is also more compatible with the field of view of a multibeam obstacle avoidance sonar. By constraining the search domain of a path planner to more relevant and feasible solutions, the search efficiency of the planner can be improved significantly.

The application of constraints in the objective function of a path planner is required for any discretisation approach. There are several methods proposed to handle constraints in the objective function of metaheuristic optimization algorithms. For PSO-based algorithms, the two most suitable methods of constraint handling are (Coath & Halgamuge, 2003):

1. Penalty function method that uses soft constraints.
2. Rejection of infeasible solutions method that uses hard constraints.

When using penalty functions in a minimization problem, penalties are added to the fitness value of a solution if the soft constraints are violated. Thus, the objective function requires

the fitness values and the penalties to be minimized simultaneously. The benefit of using a penalty function is that the soft constraint does not need to be satisfied in every iteration; instead, it can be optimized over the iterations. This reduces the solution generation time in every iteration during the optimization (Dariani et al., 2014). However, an improper penalty function may lead to premature convergence. A penalty function must be developed delicately to maintain a balance between preserving feasibility and achieving optimality.

Hard constraints must be satisfied by all solutions. Infeasible solutions that violate a hard constraint are rejected. When using hard constraints in PSO-based algorithms, all particles are forced into the feasible domain by regenerating infeasible particles during the initialisation of the algorithms. This may lead to a longer initialisation process, especially when the feasible domain is small. In every iteration of the algorithm, an infeasible particle also requires to be regenerated into the feasible domain. Consequently, the computational time of the algorithm may increase when using a hard constraint.

The effectiveness of constraint-handling methods is fully dependent on the optimization problem (Michalewicz, 1995). AUV path planning is a multi-objective non-linear optimization problem that involves multiple conflicting criteria and constraints. The optimal combination of constraint settings for a path planner should maintain a balance between its solution optimality, feasibility and computational requirements.

4.2 Problem Formulation

This section describes the formulation of the AUV path planning problem by using the polar coordinate system and different combinations of constraints. The problem space was discretised using concentric circles/spheres because it is more compatible with a 3D search space, which can allow more information from the operational environment of an AUV to be embodied during the path planning process.

4.2.1 Path Formulation

A path solution can be represented as a particle in PSO-based algorithms. The entries of a particle's position vector represent the coordinates of the path nodes. Given that every path comprises n+2 nodes (including the starting and end points), the path optimization process involves n nodes. The position vector of a 2D particle requires 2n dimensions to register the polar coordinates of n node(s); this includes n dimension(s) for radial coordinates r and n

dimension(s) for azimuthal angle coordinates Φ. A 3D particle requires $3n$ dimensions to record n node(s) in the spherical coordinates, which include extra n dimension(s) for polar angle coordinates Θ. The position vector X of the i^{th} particle in 2D and 3D can be given by Equations (2.35) and (2.36), which correspond to the polar/spherical coordinates in Equation (4.1).

$$\begin{aligned} r_i^t &= \left[\chi_{i,1}^t, \ldots, \chi_{i,j}^t, \ldots, \chi_{i,n}^t \right] \\ \Phi_i^t &= \left[\chi_{i,n+1}^t, \ldots, \chi_{i,n+j}^t, \ldots, \chi_{i,2n}^t \right] \\ \Theta_i^t &= \left[\chi_{i,2n+1}^t, \ldots, \chi_{i,2n+j}^t, \ldots, \chi_{i,3n}^t \right] \end{aligned} \tag{4.1}$$

The Cartesian coordinates of the j^{th} path node recorded by the i^{th} particle can be obtained from the polar coordinates by using Equation (4.2) and from spherical coordinates by using Equation (4.3).

$$\begin{aligned} x_{i,j} &= r_{i,j} \cos \Phi_{i,j} \\ y_{i,j} &= r_{i,j} \sin \Phi_{i,j} \end{aligned} \tag{4.2}$$

$$\begin{aligned} x_{i,j} &= r_{i,j} \cos \Phi_{i,j} \sin \Theta_{i,j} \\ y_{i,j} &= r_{i,j} \sin \Phi_{i,j} \sin \Theta_{i,j} \\ z_{i,j} &= r_{i,j} \cos \Theta_{i,j} \end{aligned} \tag{4.3}$$

The path nodes were connected to form a path by using the B-spline geometry based on the basis function in Equation (2.37). The fitness of the optimal path can be given by the aggregate objective function in Equation (2.40). The main objective function was developed to measure the travel time of the AUV by using Equation (2.41), while other subsidiary functions were designed according to the constraint settings described in the next section.

4.2.2 Constraint Settings

Hard and soft constraints were used in the objective function to generate a smooth, feasible and collision-free path that can satisfy the following objectives and boundaries:

- Obstacle avoidance: Avoid collision and keep a safe distance from obstacles.
- Radial boundary: Control the placement of path nodes.
- Azimuthal boundary: Ensure the path satisfies the minimum turning radius.
- Polar boundary: Ensure the path satisfies the pitch control limitation.

Different combinations of hard and soft constraints were applied and compared for their effects on the performance of a path planner. Table 4.1 outlines the constraint settings of the test cases investigated in this chapter.

Table 4.1: Constraint settings of test cases.

Objectives	**Test case**			
	HBHO	HBSO	SBHO	SBSO
Obstacle avoidance	Hard	Soft	Hard	Soft
Radial, azimuthal & polar boundaries	Hard	Hard	Soft	Soft

For the cases of hard constraints, an infeasible particle that violates the constraints will be regenerated. Meanwhile, when soft constraints are violated, the fitness of the particle will be penalized according to a penalty function. Regardless of the type of constraint used for obstacle avoidance, a path's exposure to obstacles requires to be measured according to the procedures outlined in Section 2.2.2 to ensure a collision-free path. When using a hard constraint, the feasibility of a path solution was checked by using Equations (2.51) and (2.52). The hard constraint is satisfied if a path does not intersect with obstacles. When a soft constraint is used, the penalty function in Equation (2.54) was used to penalise a solution if its path intersects with obstacles. The penalty for violating the soft constraint was proportional to the length of the path segment contained in the obstacle space.

To improve the search efficiency of the path planner, the placement of path nodes was controlled by a radial boundary. Each path node was constrained to lie within a concentric annulus, which is the region bounded by a pair of adjacent concentric circles/spheres with different radii. The radii were pre-defined by a lower boundary $R_{\min}$ and an upper boundary $R_{\max}$ as shown in Equation (4.4).

$$\begin{aligned} R_{\min} &= \left[0, r_d, 2r_d, \ \ldots\ , r_{target}\right] \\ R_{\max} &= \left[r_d, 2r_d, 3r_d, \ \ldots\ , r_{target}\right] \end{aligned} \tag{4.4}$$

where r_d is the radial distance between two concentric circles and r_{target} is the radial coordinate of the target location. The total number of path nodes n required for generating the path can be controlled by r_d as defined by Equation (4.5), which is rounded up towards positive infinity.

$$n = \left\lceil \frac{r_{target}}{r_d} \right\rceil \tag{4.5}$$

When using a hard constraint, a path solution is deemed infeasible if any of its path nodes exceed the boundaries $R_{\min}$ or $R_{\max}$. For the cases with soft-constrained boundaries, the following penalty function F_3 was applied.

$$F_3(X_i) = \sum_{j=1}^{n} \begin{cases} 0 & , \quad \text{if } R_{\min,j} \geq r_{i,j} \geq R_{\max,j} \\ R_{\min,j} - r_{i,j} & , \quad \text{if } r_{i,j} < R_{\min,j} \\ r_{i,j} - R_{\max,j} & , \quad \text{if } r_{i,j} > R_{\max,j} \end{cases} \tag{4.6}$$

To ensure path solutions comply with the minimum turning radius and pitch limitation of an AUV, the search domain of azimuthal angle coordinates and polar angle coordinates were constrained by using an azimuthal boundary Φ_{max} and a polar boundary Θ_{max}, respectively. A feasible path solution that satisfies the hard constraints must have all its path nodes fulfilling the conditions of $|\Phi_{i,j}| < \Phi_{max}$ and $|\Theta_{i,j}| < \Theta_{max}$. When using soft constraints, the penalty costs followed the functions in Equations (4.7) and (4.8).

$$F_4(X_i) = \sum_{j=1}^{n} \begin{cases} 0 & , \quad \text{if } |\Phi_{i,j}| \leq \Phi_{\max} \\ |\Phi_{i,j}| - \Phi_{\max} & , \quad \text{if } |\Phi_{i,j}| > \Phi_{\max} \end{cases} \tag{4.7}$$

$$F_5(X_i) = \sum_{j=1}^{n} \begin{cases} 0 & , \quad \text{if } |\Theta_{i,j}| \leq \Theta_{\max} \\ |\Theta_{i,j}| - \Theta_{\max} & , \quad \text{if } |\Theta_{i,j}| > \Theta_{\max} \end{cases} \tag{4.8}$$

4.3 Numerical Simulations

2D and 3D path planning scenarios were simulated to evaluate the effect of constraint-handling methods on the performance of the path planner. The analysis was based on the Monte Carlo method with 1000 simulation runs. The SDEQPSO and SDEAPSO algorithms proposed in Chapter 3 and the QPSO algorithm were used in the path planners with a population size of 150 particles. The problem space was an ocean field of 500×500 square metres for 2D and 500×500×500 cubic metres for 3D. Non-uniform ocean currents and static obstacles of different sizes were present in the problem space. The current field was generated based on the data obtained from the field experiment conducted at Beauty Point, Tasmania, Australia.

The AUV was required to travel with a preset water-referenced velocity of 1.15 m/s. The buffer distance for obstacle avoidance was set to 1 metre based on the properties of the REMUS 100 AUV. For the radial boundary, r_d was set to 50 metres. The azimuthal boundary $\Phi_{\max}$ was set to ±60° because typical forward-looking sonars have a field of view of 120°. The polar boundary $\Theta_{\max}$ was set to ±15° as the pitch angle of a torpedo-shaped AUV rarely exceeds 20° during an operation. The simulations were conducted on a machine with Intel Core i5-6300U CPU @ 2.4GHz with 8GB RAM.

In addition to the test cases described in Table 4.1, test cases using an unconstrained objective function (uncon.) were also included for comparison purposes. The unconstrained cases were configured by using the formulation in Section 2.2, which employed a grid-based approach with the Cartesian coordinate system. It was discovered that the runtime required by the SDEAPSO path planner was too high when a hard constraint was used for obstacle avoidance. This can be explained by the nature of the position and velocity update equations in the SDEAPSO algorithm. Unlike QPSO and SDEQPSO that use the swarm's mean best position in their update equations, SDEAPSO has a stronger cognitive component in its equations. When the feasible space that can satisfy the hard constraint of obstacle avoidance is small, an infeasible solution may require an impractically long time to move back into the feasible space before the iteration can continue. Thus, the test cases of HBHO and SBHO for SDEAPSO were excluded from the analysis.

4.3.1 Comparison of Constraint Settings

The performances of the path planners using different combinations of constraint settings were compared based on their solution qualities, stabilities, and computational requirements. These properties can be shown by the fitness values of the solutions obtained and the runtime required to obtain the solutions. A lower fitness value is desirable for the path planning problem.

Based on the Shapiro-Wilk normality test with a significance level of 0.05, the simulation results were not normally distributed. Hence, the following analysis used non-parametric boxplots that show the medians and interquartile ranges of the solutions to evaluate the searching ability and stabilities of the path planners. Figure 4.1 and Figure 4.2 depict the boxplots of fitness values obtained in different test cases under 2D and 3D scenarios, respectively.

In the boxplots, the black whiskers indicate the data ranges while the blue boxes show percentile ranges. The red lines inside the boxes represent the medians of the fitness values. The Kruskal-Wallis test, which is a non-parametric ANOVA, was used with a significance level of 0.05 to rank the path planners based on their obtained fitness values. The ranking procedure used the Holm-Bonferroni stepdown approach. The algorithms were given the same rank if they are not statistically different from one another. Table 4.2 summarises the obtained results, including the ANOVA ranks (R), the medians of runtime (T), and the total ranks ($\#R$). The total ranks were given by the summation of the ranks in the 2D and 3D scenarios. The impact of algorithm runtime was not considered in the ranking procedure.

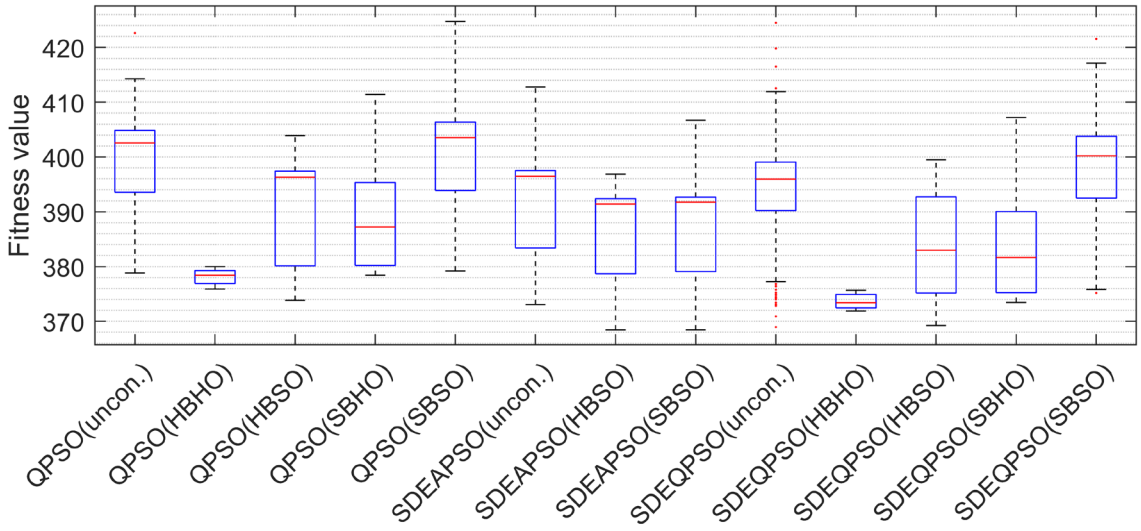


Figure 4.1: Fitness values obtained in 2D scenario.

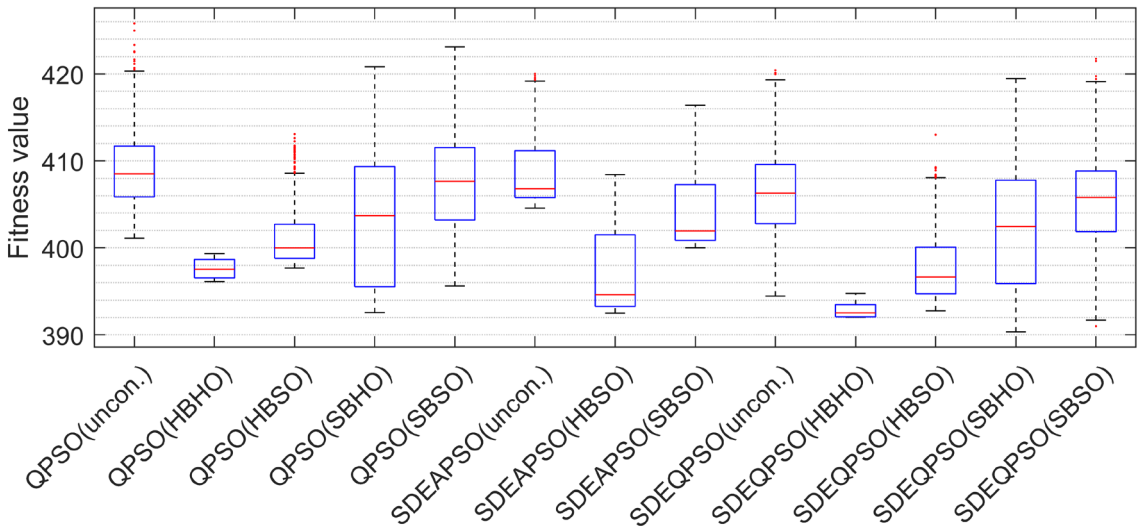


Figure 4.2: Fitness values obtained in 3D scenario.

Table 4.2: Simulation results for comparison of constraint settings.

Algorithm	Case	2D		3D		#***R***
		R	*T(s)*	*R*	*T(s)*	
QPSO	Unconstrained	11	11.7	12	24.6	23
	HBHO	2	38.4	2	46.7	4
	HBSO	7	12.4	5	26.7	12
	SBHO	7	20.8	7	35.3	14
	SBSO	13	11.9	11	24.6	24
SDEAPSO	Unconstrained	9	11.9	12	25.1	21
	HBSO	5	15.2	2	26.1	7
	SBSO	5	11.9	8	25.1	13
SDEQPSO	Unconstrained	10	11.5	10	23.6	20
	HBHO	1	38.4	1	46.9	2
	HBSO	3	12.1	2	25.1	5
	SBHO	3	20.6	5	34.7	8
	SBSO	11	11.7	8	24.0	19

Based on Figure 4.1, Figure 4.2 and Table 4.2, the HBHO case of the SDEQPSO algorithm achieved the top rank in both 2D and 3D scenarios. This was followed by the HBHO case of the QPSO algorithm. These HBHO cases also produced the highest stabilities, as shown by their relatively lower interquartile ranges in the boxplots. Although the HBHO case of the SDEAPSO algorithm was inadequate for comparison, the HBHO setting was observed to have the best performance in terms of solution qualities compared to other settings. However, by comparing the runtime, it was found that the HBHO setting had the highest computational requirement in both 2D and 3D scenarios.

The HBSO cases of SDEQPSO and SDEAPSO were ranked third and fourth, respectively. The HBSO setting can maintain a good balance between the searching ability and computational requirement of the path planner, thus indicating a higher search efficiency. Although the SBHO case of SDEQPSO achieved a similarly good solution quality in 2D and 3D, it required a much higher runtime. Based on this observation, it can be deduced that the hard constraint setting for obstacle avoidance was the main reason for the undesirable increase in computational requirement. When a hard constraint was used, an infeasible

solution was forced to move back into the feasible space by regeneration, which led to the extra time required for computation. By making an algorithm-wise comparison, it was found that the SDEQPSO has the best overall performance, although SDEAPSO has better performance in the SBSO case.

4.3.2 Vehicle Path Verification

To verify the generated vehicle paths, the 2D and 3D solutions of the SDEQPSO path planner with the HBSO setting were used to produce a reference trajectory for the REMUS 100 AUV model, which was described in Section 3.7. The feasibility of the paths was first checked for their compliances with the vehicular constraints of the REMUS 100 AUV. As shown in Figure 4.3, the curvature of the planned paths was well above the minimum turning radius of the vehicle.

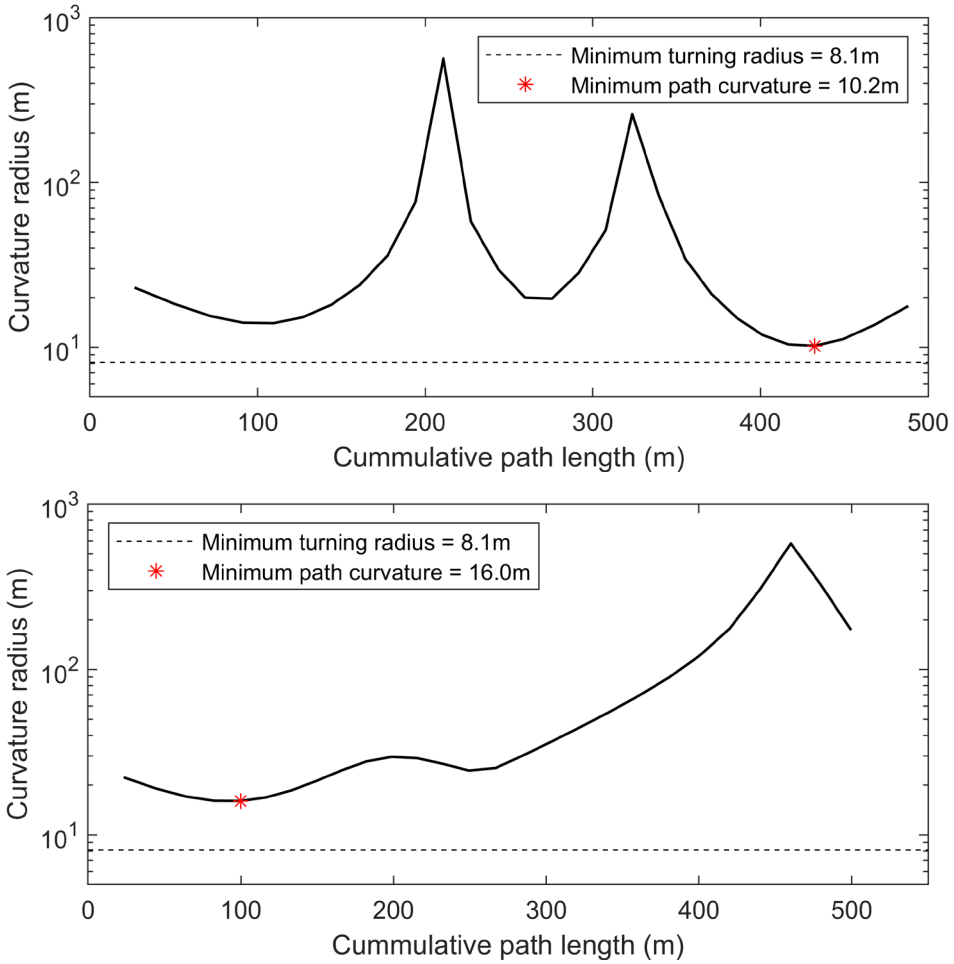


Figure 4.3: Curvature radius of planned paths for 2D (top) and 3D (bottom).

Next, the cross-track errors of the simulated paths relative to the planned paths for the 2D and 3D scenarios were compared and graphed in Figure 4.4. It can be observed that the vehicle was able to follow the planned paths closely in the simulations, with cross-track errors of well below 1 metre in both scenarios. Therefore, the simulation results showed that the path solutions generated by the path planner were smooth and feasible for the path planning application.

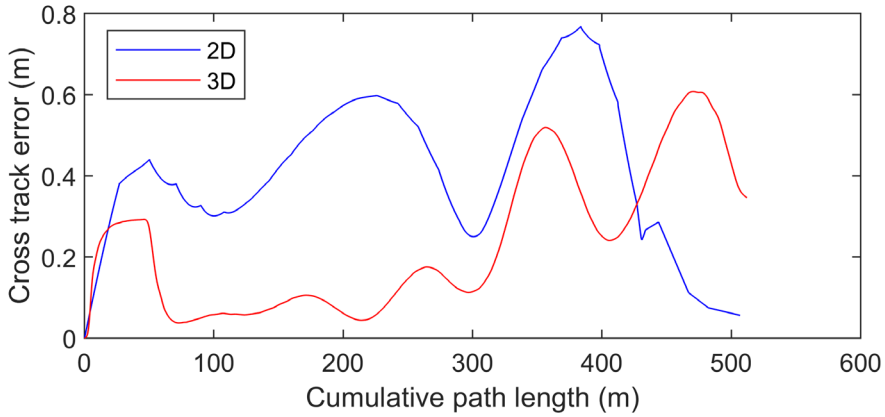


Figure 4.4: Cross-track error of simulated paths relative to planned paths.

4.4 Chapter Summary

This chapter evaluated the performance of an AUV path planner under different combinations of constraint settings. The SDEQPSO path planner with the hard constraint setting for boundary conditions and the soft constraint setting for obstacle avoidance produced the best performance in terms of search efficiency, as shown by its higher solution quality and lower computational requirement. The path planners that used a hard constraint for obstacle avoidance were found to have significantly higher computational requirements. The proposed path planner generated a safe and dynamically feasible path for the REMUS 100 AUV, which was verified through the simulation of an AUV dynamic model. The optimal constraint setting determined in this chapter successfully improved the search efficiency of the path planner, which is of crucial importance for developing an online path planner in the next chapter.

This page is intentionally left blank.

Chapter 5.

Online Path Replanning in Unknown Dynamic Environments

This chapter[4] presents an online AUV path planner that employed a path replanning approach and the SDEQPSO algorithm to optimize an AUV mission in an unknown, dynamic and cluttered ocean environment. Without requiring any system model or prior knowledge of the environment, the proposed path replanner can generate a time-optimal path based on the onboard sensor data. Different configurations for the FLS and H-ADCP sensors were considered in 2D and 3D simulations. The SDEQPSO path replanner was found to be capable of generating a time-optimal path that offered up to a 13% reduction in travel time compared to the situation where the vehicle simply followed a path with the shortest distance. The proposed replanning technique also showed consistently better performance over a reactive path planner in terms of solution quality, stability, and computational efficiency. The robustness of the replanner was verified under stochastic processes using the Monte Carlo method. The planned path fulfilled the vehicle's safety and physical constraints, while intelligently exploiting ocean currents to improve the vehicle's efficiency.

5.1 Path Replanning

To date, significant research has been conducted to study path planning of AUVs, especially for operations in *a priori* known or static environments. Nevertheless, path planning of an AUV operating in an unexplored, dynamic, and cluttered underwater environment remains a computationally intractable problem. An online AUV path planner must be able to

[4] This chapter was modified from the following publication: Lim, H. S., Chin, C. K. H., Chai, S., & Bose, N. (2020). Online AUV path replanning using quantum-behaved particle swarm optimization with selective differential evolution. *Computer Modeling in Engineering & Sciences*, 125(1), 33-50.

continuously generate a safe and feasible path in real-time by adapting to unexpected changes in the environment. Online path planning requires a computationally efficient algorithm because the allowable time for planning is often limited during an AUV operation. A practical online planner should establish a balance between its path quality and computational cost.

To date, various techniques were proposed to perform online path planning. An intuitive approach known as local path planning (Benjamin et al., 2019; Casalino et al., 2009; Larson et al., 2006; Sun et al., 2018; Yao et al., 2020) can generate a safe vehicle path effectively by making local adjustments while following a previously planned path. This approach ensures the safety of a path by sacrificing a certain degree of path optimality.

Another approach known as reactive path planning (Belkhouche & Bendjilali, 2012; Candeloro et al., 2017; Naeem et al., 2012; Petres et al., 2011; Singh et al., 2018; Vasile & Belta, 2014) generates a new path reactively to adapt to the varying environment while previously planned paths are discarded. Reactive planning approach was also applied with neural network and reinforcement learning to generate safe AUV paths in dynamic environments (Cheng & Zhang, 2018; Cui et al., 2017; Duguleana & Mogan, 2016; Lin et al., 2019). The implementations of these learning algorithms in actual AUVs are challenging because it is expensive and time-consuming to train a generalised model that is suitable for real operations.

On the other hand, path replanning is a technique that generates a path by reusing the previous solution(s) to improve computational efficiency. Instead of starting afresh in every planning cycle, a path replanner modifies solution(s) from the previous planning cycle to generate a new path efficiently. Path replanning can be performed by using a single previous solution (Galceran et al., 2015; Hernández et al., 2019; Ma et al., 2018; Park et al., 2013; Sun & Zhu, 2016) or a pool of previous solutions (Biswas et al., 2016; Hernández et al., 2011; Lv et al., 2019; MahmoudZadeh et al., 2018; Zeng et al., 2015; Zhou et al., 2018).

The SDEQPSO algorithm is a population-based optimization algorithm that is suitable for the application of path replanning from multiple previous solutions. It can maintain the entire population of previous solutions, which can be used for replanning the path at any time throughout the mission.

In this chapter, the path replanning scheme was employed to handle an online path planning scenario in a fully unknown and dynamic ocean environment. The path replanning process was carried out online and continuously at an adaptive interval while the AUV navigates towards its target. The adaptive replanning interval was devised to be reactive to surrounding environments, meaning that a previously planned path is replanned when it is unsafe or less optimal due to environmental changes.

The flags that can trigger path replanning are:

- Elapsed time since the previous plan exceeds a preset limit.
- Newly detected obstacles show up within the safe zone of the vehicle.
- Newly detected obstacles intersect the previously planned path.

In order to ensure the planned path was optimized for a spatiotemporal current field, the first replanning flag enabled the path replanner to refine the path periodically even when no conflicting obstacles were detected. The elapsed time limit was defined to be inversely proportional to the vehicle and current velocities. Based on the vehicle water-referenced velocity V_a and the maximum current velocity $V_{c,max}$ measured by a current profiler, the elapsed time limit t_{replan} can be given as below:

$$t_{replan} = \frac{D_{adcp}}{V_a + V_{c,max}} \tag{5.1}$$

where D_{adcp} is the effective range of the vehicle's current profiler.

When replanning is required, the solutions of the SDEQPSO path replanner from the previous planning cycle can serve as the initial solutions for path optimization to improve the search efficiency. The replanner modifies its previous solutions to generate a new path that is optimized for its new planning conditions. The process of reusing the previous solution for path replanning is described in Figure 5.1, in which the grey dots and black circles represent the population of path nodes that can be used to construct an AUV path. During a new planning cycle, a portion of the previous solutions (grey dots) can be retained and used as the initial population (black circles) for replanning the path.

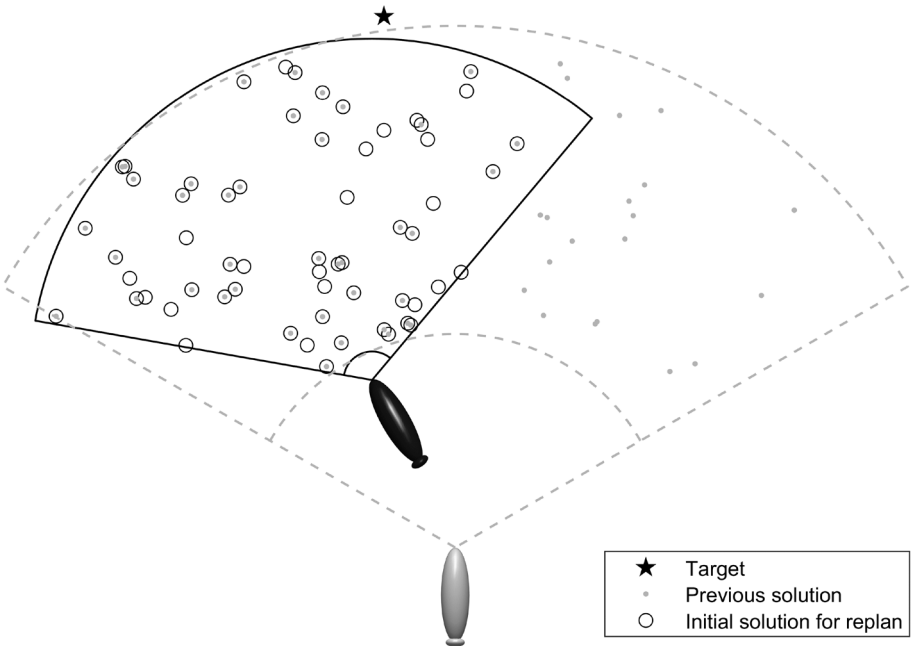


Figure 5.1: Reuse of solutions in path replanning process (grey vehicle denotes the previous position, and black vehicle denotes the current position).

The initialisation of path replanning begins with defining the new boundary conditions based on a new starting point, which is the AUV's current position. The path nodes behind the new starting point are removed. Next, solutions that satisfy the new boundary conditions are retained, while solutions that violate the boundary conditions are rejected and regenerated. After the initialisation process, the solutions undergo the SDEQPSO iteration to determine the optimal path. The proposed path replanner can be executed according to the following pseudocode.

Step 1. **Define** the algorithm parameters and ocean environments.

Step 2. **Initialise** a group of candidate paths by generating particles with random positions in Equation (2.1). Set *pbest* to the current particle positions.

Step 3. **Check** the path replanning flag.

Step 4. **If** replanning flag == TRUE

Define boundary conditions for r, Φ, and Θ.

Check the feasibility of particles for the boundary conditions.

Regenerate infeasible particles with random positions in Equation (2.1).

While the stop criteria are not met,

For $t = 1, 2, \ldots, t_{max}$,

Evaluate particle fitness $F(X_i^t)$ using the objective function F.

Update *pbest* and *gbest* using Equations (2.4) and (2.5) respectively.

Update *mbest* by using Equation (2.10).

Update β using Equation (2.12).

For each particle $i = 1, 2, \ldots, N$,

Update particle position using Equation (2.11).

End

Sort all particles according to their personal best fitness.

For $k = 1, 2, \ldots, N_s^{th}$ best performing particle,

Mutation: Generate mutated vector U_k^t using Equation (2.25).

Crossover: Generate trial vector T_k^t using Equation (2.26).

Natural selection: Replace k^{th} worst-performing particle with T_k^t.

End for

End for

End while

Return *gbest* that contains the optimal path upon algorithm termination.

Else

Follow the previous path.

End if

Step 5. **Back** to Step 3 if the mission is not completed.

5.2 AUV Simulation Model

The simulation of an online path planning scenario requires the use of an AUV mathematical model. The SDEQPSO path planner can continuously generate an AUV path based on the feedback from the sensors and the AUV dynamic model as illustrated in Figure 5.2. The planned paths were used to generate a reference trajectory for the dynamic model of the Hydroid REMUS 100 AUV. The AUV model and path following controller described in Section 3.7 were used in the simulation. The models of the FLS and H-ADCP sensors are outlined in this section.

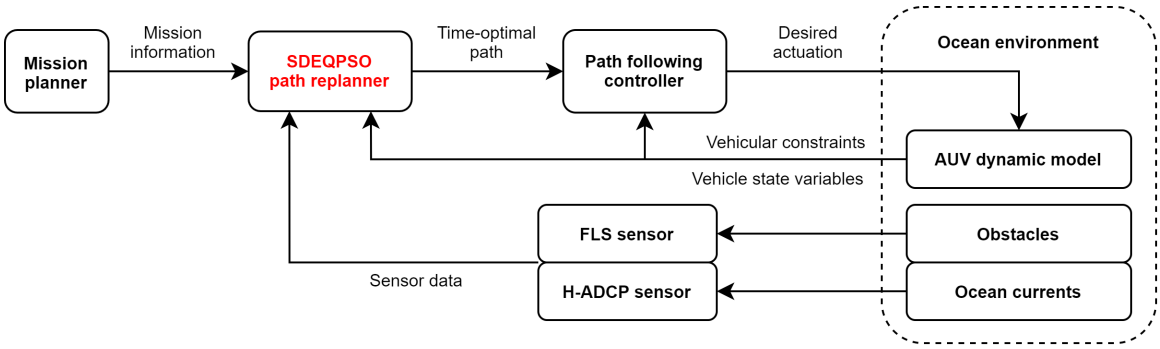


Figure 5.2: Implementation of SDEQPSO path replanner.

5.2.1 Forward-looking Sonar Model

A forward-looking sonar (FLS) model was used in the simulation for the detection of obstacles. Based on the specification of the REMUS 100, the settings of the FLS model were configured as follows: 80 metres detection range, 120° field of view, 121 number of beams (with 1° separation between beams), and 100Hz detection frequency.

Generally, the sonar configuration of an AUV can vary depending on the mission requirements. The horizontal sonar configuration, in which the fan-shaped FLS model was installed in such a way that the sonar fan aligns with the horizontal plane of the vehicle, is suitable for missions such as area coverage survey. Some missions such as operations underneath ice shelves or near-seabed operations require the FLS to be configured in the vertical plane. Therefore, two sonar configurations were considered in this study as shown in Figure 5.3. The vertical sonar configuration has an offset of 20° above the horizontal plane.

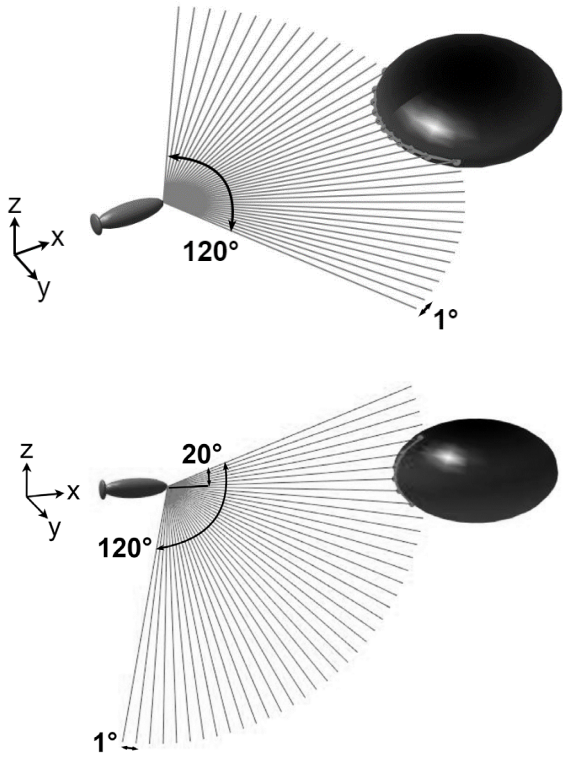


Figure 5.3: FLS sensors configured in horizontal (top) and vertical (bottom) planes.

During the simulation, all obstacles in the problem space were configured to be irregular and *a priori* unknown. The FLS model can detect the boundaries of obstacles and generate sensor measurements based on the coordinates of the detection points. The sensor data were recorded by the path planner to achieve obstacle avoidance during path computation. The path planner can maintain a list of relevant detection points by removing irrelevant detection points based on their distances to the vehicle.

5.2.2 Current Profiler Model

In order to enable the adaptation of path solutions to ocean currents, the path planner was provided with continuous information of current profiles based on the simulated measurements from a forward-looking horizontal acoustic Doppler current profiler (H-ADCP). These sensor measurements were used by the path planner to adapt its solutions and generate a time-optimal path. The path planning simulation used a 300kHz H-ADCP with 200 metres detection range, which is able to reconstruct a velocity profile of 200 metres × 50 metres in the forward-looking region of the vehicle (Garau et al., 2006).

5.3 Problem Formulation

The objective functions of the path planner were formulated according to the descriptions in Section 4.2. The optimal constraint settings established in Chapter 4 were applied. In order to improve the search efficiency during path replanning, the placement of the path nodes was controlled by the radial boundary using a hard constraint. Path nodes were constrained to lie within a lower boundary $R_{\min}$ and an upper boundary $R_{\max}$. Hard constraints were also applied to ensure the path solutions respect the minimum turning radius and the pitch limitation of the AUV. An azimuthal boundary Φ_{max} and a polar boundary Θ_{max} were used to constrain the search domain of azimuthal angle coordinate and polar angle coordinate. A path solution fulfils the constraints if $|\Phi_{i,j}| < \Phi_{max}$ and $|\Theta_{i,j}| < \Theta_{max}$; otherwise, the solution will be regenerated.

5.3.1 Obstacle Avoidance Using Combined Constraint

It was found in Chapter 4 that the hard constraint setting for obstacle avoidance can generate an excellent solution quality. However, a hard constraint may lead to an undesirable increase in the computational cost of a path planner due to the additional runtime required to regenerate infeasible solutions. A soft constraint has a lower computational requirement because it can be optimized over time. Although the use of a soft constraint can help to maintain a good balance between a path planner's solution quality and computational cost, a vehicle may bump into obstacles if its path is soft-constrained to an inadequate distance with obstacles. This may occur in practice due to the vehicle's actuation limitations or external forces, such as ocean currents.

Therefore, a combined constraint approach was developed and applied for obstacle avoidance to ensure a collision-free path while maintaining a reasonable computational load. In this approach, two parameters were used to control the obstacle avoidance behaviour:

1. A buffer distance d_{buff}, which was applied with a soft constraint.
2. A safety distance d_{safe}, which was applied with a hard constraint.

The buffer distance allows the vehicle to bump into the buffered obstacle in real AUV operations. It was used to address the concerns for sensor uncertainties in FLS measurements, as well as the limitations in vehicular actuation and the effect of external forces on the vehicle. On the other hand, the safety distance must be maintained by the

vehicle to avoid collisions during an operation. In this combined constraint approach, the soft constraint was configured to be stricter than the hard constraint. Thus, it reduces the tendency of solutions to violate the hard constraint, subsequently lowering the computational load of the path planner.

To achieve obstacle avoidance using the combined constraint approach, the algorithm needs to compute the minimum distance between a path and a detected obstacle, which is represented as detection points generated by the FLS sensor. Assuming a solution X_i in a 3D Euclidean space, the problem involves an obstacle h with a detection point $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and a path segment that connects two adjacent waypoints $p_{i,j}$ and $p_{i,j+1}$. Firstly, the nearest point on the segment $p_{i,j}\,p_{i,j+1}$ to the point $O_{c,h}$ was determined. The nearest point $\rho(s)$ can be parameterized by Equation (5.2). Next, vector projection was used to obtain Equation (5.3).

$$\rho(s) = p_{i,j} + s\left(p_{i,j+1} - p_{i,j}\right) \tag{5.2}$$

$$\hat{s} = \frac{\left\langle O_{c,h} - p_{i,j}\ ,\ p_{i,j+1} - p_{i,j} \right\rangle}{\left\| p_{i,j+1} - p_{i,j} \right\|^2} \tag{5.3}$$

To find the parameter s that gives the minimum distance within the path segment, Equation (5.4) was used to clip s to the range of [0,1]. Thus, the minimum distance from $O_{c,h}$ to $\rho(s)$ can be determined by using Equation (5.5), which gives the obstacle distance D_{obs} between the path segment and the detected obstacle.

$$s' = \min\left(\max\left(\hat{s}, 0\right), 1\right) \tag{5.4}$$

$$D_{obs}(s) = \left\| \rho\left(s'\right) - O_{c,h} \right\| \tag{5.5}$$

A hard constraint was applied to D_{obs} by using the safety distance d_{safe}. A feasible path must maintain a distance greater than d_{safe} from all obstacles. A solution is deemed infeasible and will be regenerated if any of its path segments contains $D_{obs} < d_{safe}$. Additionally, the solution was soft-constrained by using the buffer distance d_{buff}. For each path segment with $D_{obs} < d_{buff}$ and $D_{obs} \geq d_{safe}$, the penalty function F_2 in Equation (5.6) can be applied to obtain the penalty cost, which is inversely proportional to D_{obs}. The total penalty of a solution can be given by Equation (5.7).

$$F_{2,h}(p_{i,j}) = \frac{d_{buff} - D_{obs}}{d_{buff}} \tag{5.6}$$

$$F_2(X_i) = \sum_{h=1}^{H} \sum_{j=1}^{m-1} F_{2,h}(p_{i,j}) \tag{5.7}$$

In dynamic and unknown environments, a detected obstacle does not necessarily remain stationary in the same location; it may move and become irrelevant over time. Moreover, the memory of the path planner for storing obstacle detections may become overloaded with time, leading to increased latency in the planner as the mission progresses. Hence, the proposed algorithm must maintain a list of relevant detection points by removing irrelevant detections based on their elapsed time and distances to the vehicle. Detections that exceeded the distance $D_{obs,max}$ or the elapsed time $t_{obs,max}$ were removed.

$$D_{obs,max} = 2 \cdot D_{fls} \tag{5.8}$$

$$t_{obs,max} = \frac{D_{obs,max}}{V_a} \tag{5.9}$$

where D_{fls} is the detection range of the vehicle's FLS sensor.

The obstacle avoidance process can be illustrated in Figure 5.4. The following pseudocode describes the combined constraint approach for achieving obstacle avoidance.

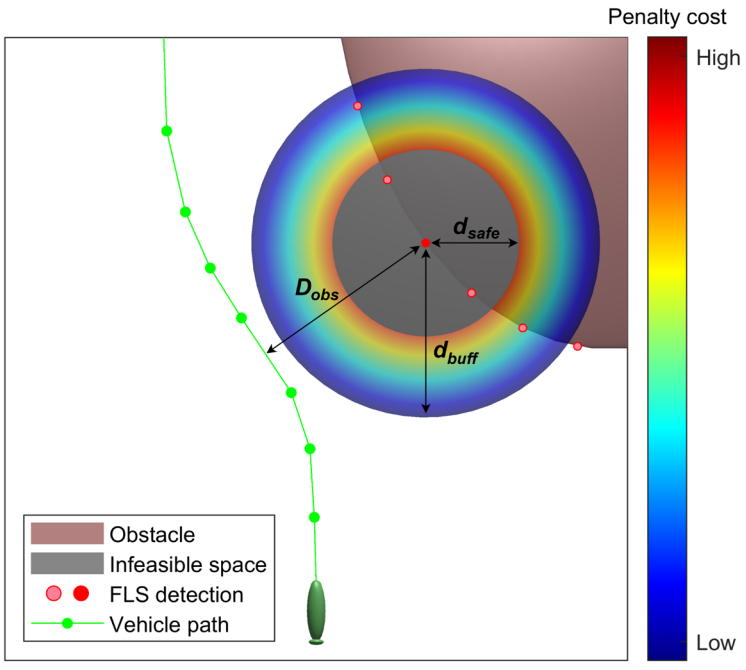


Figure 5.4: Example of obstacle avoidance scenario using an FLS detection point.

For each obstacle detection point $O_{c,h}$, where h = 1, 2, …, H,

  For each discretised waypoint $p_{i,j}$, where j = 1, …, m-1,

    Find s using Equation (5.3).

    Clip s to [0,1] using Equation (5.4).

    Find D_{obs} using Equation (5.5).

    If $D_{obs} < d_{safe}$

      Break and regenerate solutions.

    Else if $d_{safe} \leq D_{obs} < d_{buff}$

      Apply penalty using Equation (5.6).

    Else

      $F_{2,h}(p_{i,j}) = 0$

    End if

  End for

End for

5.4 Numerical Simulations

The AUV mission was simulated in 2D scenarios and subsequently 3D scenarios based on the Monte Carlo method with 1000 runs. The machine used has an Intel Core i5-6300U CPU @ 2.4 GHz with 8 GB RAM. The problem spaces of the simulations were assumed to be an underwater environment that contains 1000×1000 square metres for 2D, and 1000×1000×1000 cubic metres for 3D. *A priori* unknown obstacles and spatiotemporal ocean currents were simulated in the problem space. The placement of the *a priori* unknown obstacles was configured in such a way that they will potentially block the optimized path of the AUV. In the cases of moving obstacles, they were set to move independently in different directions at random speeds up to 0.1 m/s. The variable current field with current velocity up to 0.5 m/s was generated by using field data of ocean currents. The field data were obtained at Beauty Point, Tasmania, Australia by using the ADCP sensors of the UTAS Explorer AUV "*nupiri muka*".

The AUV was configured with a default water reference velocity of 1.15 m/s. Based on the properties of the REMUS 100 AUV, the safety distance and buffer distance required for obstacle avoidance were defined as 3 metres and 5 metres, respectively. The test cases for

the simulation are described in Table 5.1. The azimuthal boundary $\Phi_{\max}$ was set to ±60° because the FLS model had a field of view of 120°. The polar boundary $\Theta_{\max}$ was set to ±20° as the pitch angle of a torpedo-shaped AUV rarely exceeds 20° during an operation.

Table 5.1: Setups of simulation test cases.

Test Case	**Dimension**	**FLS configuration**	**Obstacles**	r_d **(m)**	Φ_{max} **(°)**	$\Theta_{\max}$ **(°)**
1	2D	Horizontal	Stationary	50	±60	-
2	2D	Horizontal	Moving	50	±60	-
3	3D	Horizontal	Stationary	50	±60	±5
4	3D	Horizontal	Moving	50	±60	±5
5	3D	Vertical	Stationary	50	±5	±20
6	3D	Vertical	Moving	50	±5	±20

During the simulation, the maximum number of iterations for the SDEQPSO algorithm was set to 100 with a pre-defined stopping threshold. This means the algorithm will be iterated up to a maximum number of 100 but will be stopped whenever the difference in solutions between iterations is less than the pre-set threshold. The population size of the SDEQPSO algorithm was set to 150 particles. The setting of algorithm parameters was based on the suggested values as discussed in Chapter 3. The performance of the path replanner was evaluated by comparison with two other path planners:

1. An SDEQPSO-based path replanner without adaptation to ocean currents. This planner neglected the velocity vector of ocean currents in Equation (2.41) and (2.42),
2. An SDEQPSO-based reactive path planner with adaptation to ocean currents. This planner reinitialized the entire particle swarm with random positions every new planning cycle.

In the Monte Carlo simulation, the robustness of the planners was assessed under scenarios with stochastic processes, *i.e.*, randomly moving obstacles and constantly changing ocean currents.

5.4.1 Performance Assessment

The solutions generated by the SDEQPSO path replanner were depicted in Figure 5.5 and Figure 5.6. In all test cases, the mission of the AUV was to traverse the ocean field towards the target while maintaining a safe distance with the obstacles and attempting to exploit the favourable currents that would assist the AUV motion. The vehicle was driven to surf the favourable currents and to avoid the adverse currents that would oppose the vehicle's motion.

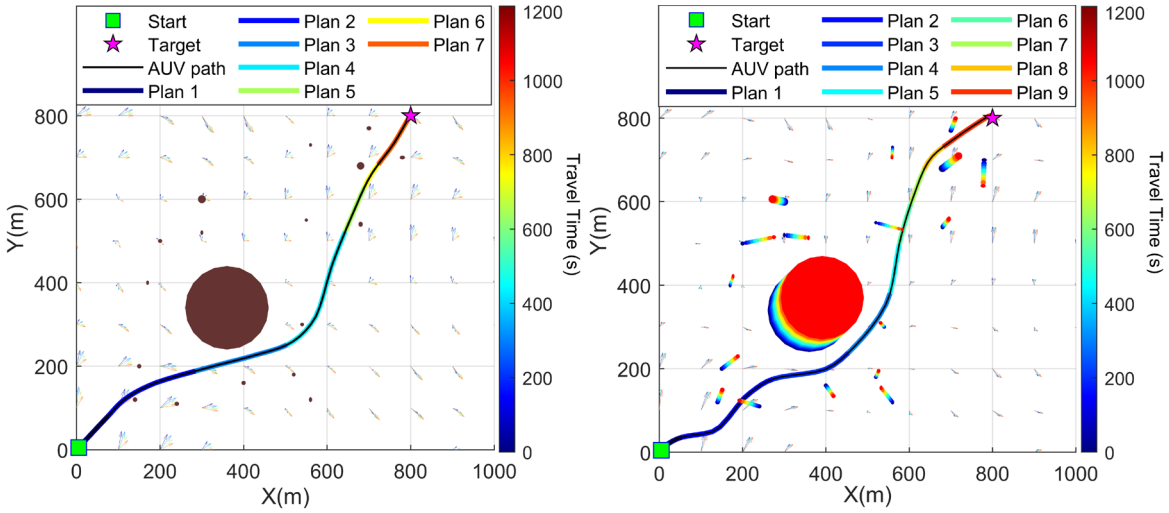


Figure 5.5: Pareto-optimal path solutions for Case 1 (left) and Case 2 (right).

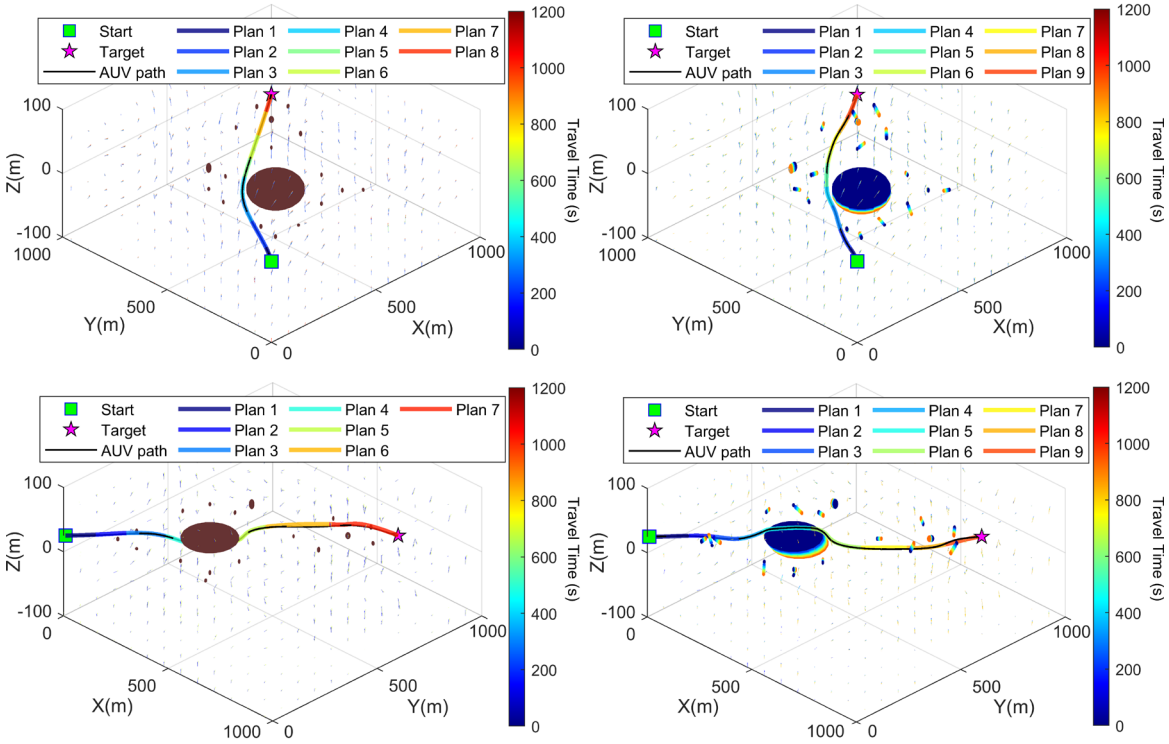


Figure 5.6: Pareto-optimal path solutions for Case 3 (top left), Case 4 (top right), Case 5 (bottom left), and Case 6 (bottom right).

The elapsed time of the AUV mission is represented by the colour bars in Figure 5.5 and Figure 5.6. Colours corresponding to the elapsed time are used for the planned path and the vector field of ocean currents. The boundaries of the static obstacles (Cases 1, 3 and 5) are coloured brown, whereas the boundaries of the moving obstacles (Cases 2, 4 and 6) are indicated by the trails coloured according to the elapsed time. Therefore, no collision will occur if the coloured path does not intersect with the brown obstacles or the obstacle trails of the same colour.

As the obstacles were intentionally placed to block the AUV path, the AUV must detour around the obstacles in every test case by replanning a new path whenever the previously planned paths collide with the obstacles detected by the FLS sensor. The vehicle with horizontal sonar configuration mainly manoeuvred by using yaw motion, whereas the vehicle with vertical sonar configuration mostly utilized pitch motion. The resultant paths are safe and collision-free as shown in Figure 5.5 and Figure 5.6. During the simulation, the AUV was able to follow the planned path closely, as shown by the executed AUV paths (black lines) that closely resemble the planned paths in all test cases.

The feasibility of the path solutions was analysed by checking against the vehicular constraints of REMUS 100. The minimum turning radius of the REMUS 100 AUV is 8.1 metres in the worst-case scenario. A feasible path must have its curvature radius greater than the AUV's minimum turning radius. As shown in Figure 5.7, the generated paths satisfied the vehicle's turning constraints.

The vehicle speed, heading and pitch of REMUS 100 when following the paths are depicted in Figure 5.8, Figure 5.9 and Figure 5.10, respectively. The AUV maintained an average speed of 1.5 m/s in all cases. The fluctuation of vehicle speed in Case 5 and 6 can be correlated to the increase in pitch motion required when using a vertical sonar configuration. Based on a conservative assumption, the REMUS 100 AUV has a pitch limitation of 20° during operations. Figure 5.10 shows that the vehicle pitch was well within the vehicular limitation. As shown in Figure 5.9 and Figure 5.10, the vehicle utilised mainly yaw motion when using the horizontal sonar configuration (Cases 1, 2, 3 and 4). When using the vertical sonar configuration (Cases 5 and 6), the vehicle manoeuvred by using mostly pitch motion. The vehicle's control forces and moments for following the planned paths in all cases can be found in Appendix D.

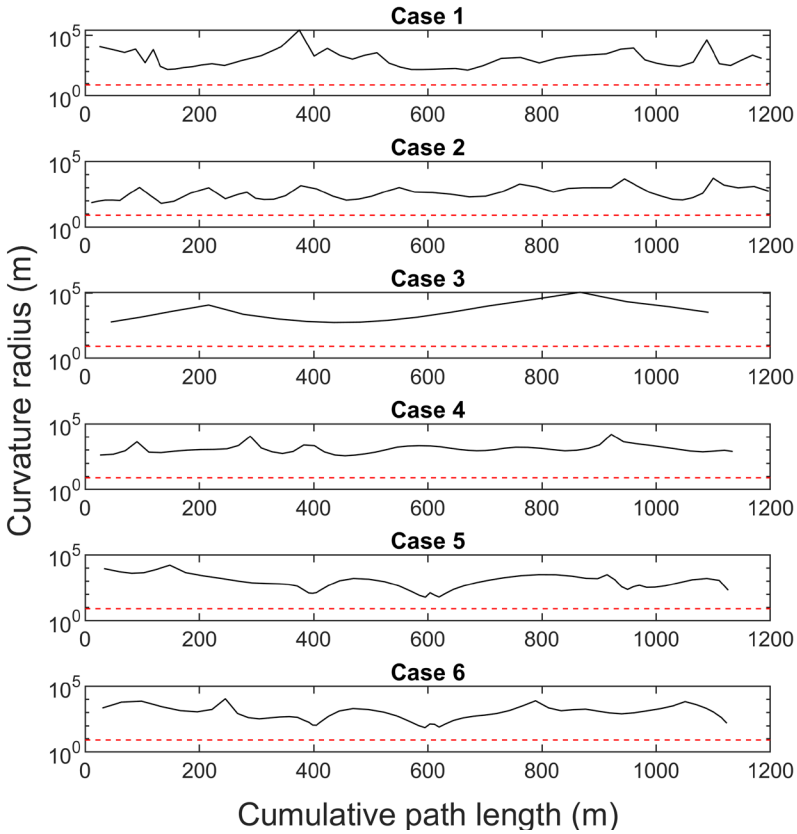


Figure 5.7: Variation of path curvature with respect to vehicular constraints (dashed line).

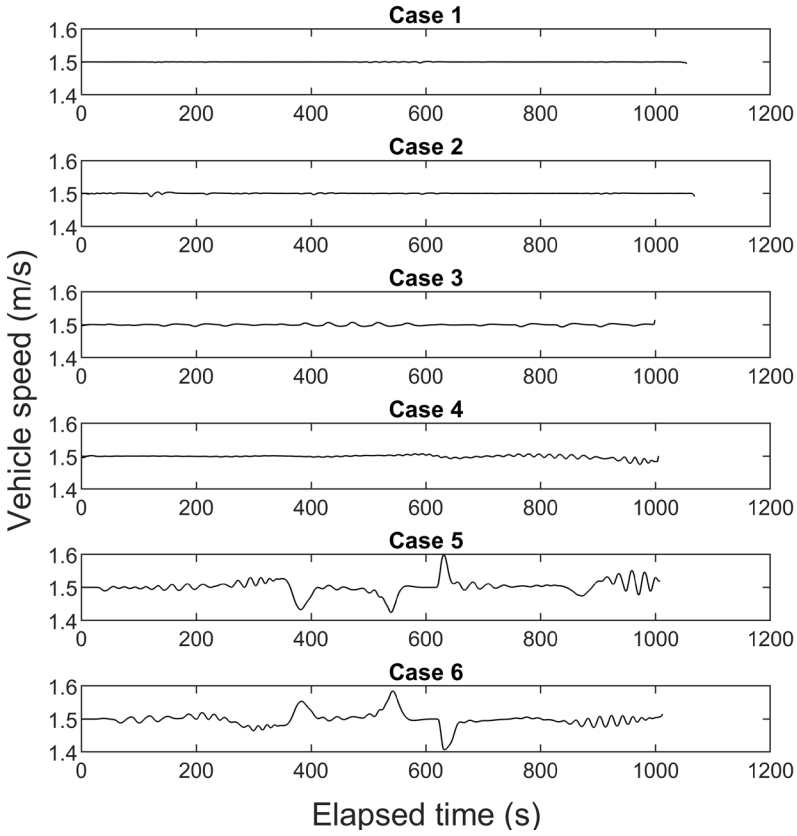


Figure 5.8: Variation of vehicle speed.

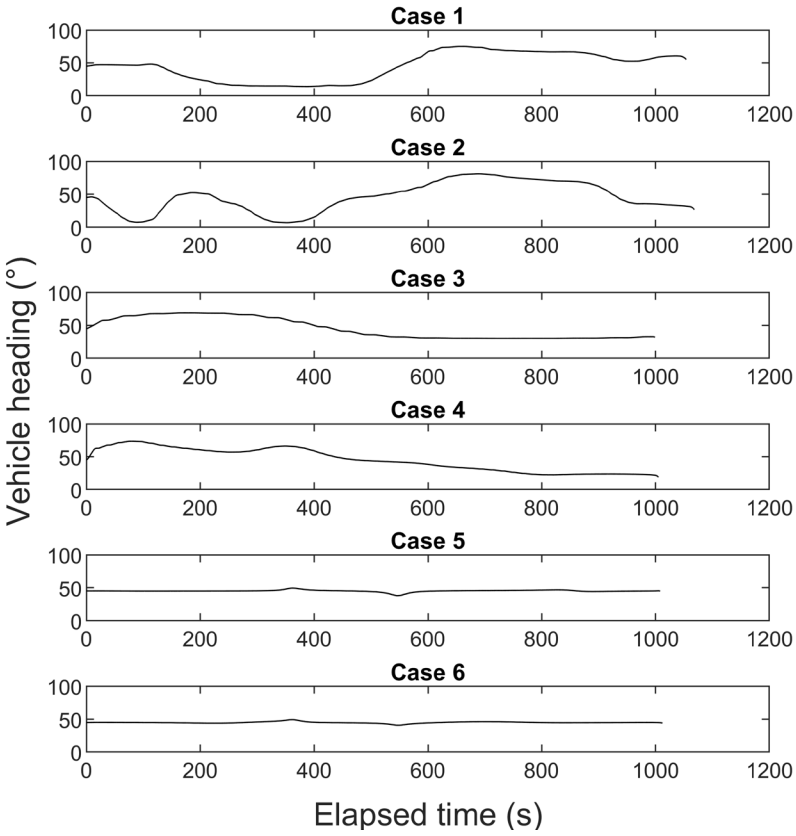


Figure 5.9: Variation of vehicle heading.

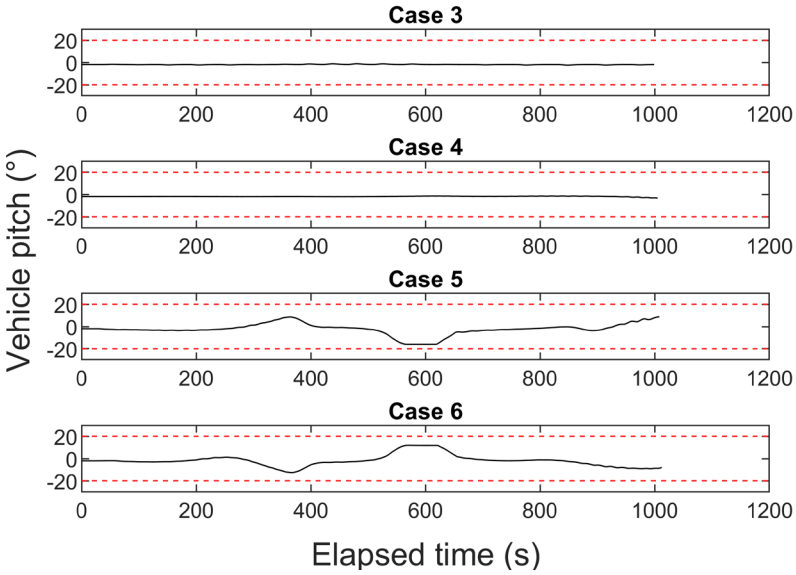


Figure 5.10: Variation of vehicle pitch with respect to vehicular constraints (dashed line).

Figure 5.11 shows the resultant cross-track errors of the executed paths relative to the planned paths during the simulations. The errors for all cases were found to be well below 1 metre (less than 0.1% of the total path length), proving that the AUV was able to follow the planned paths closely.

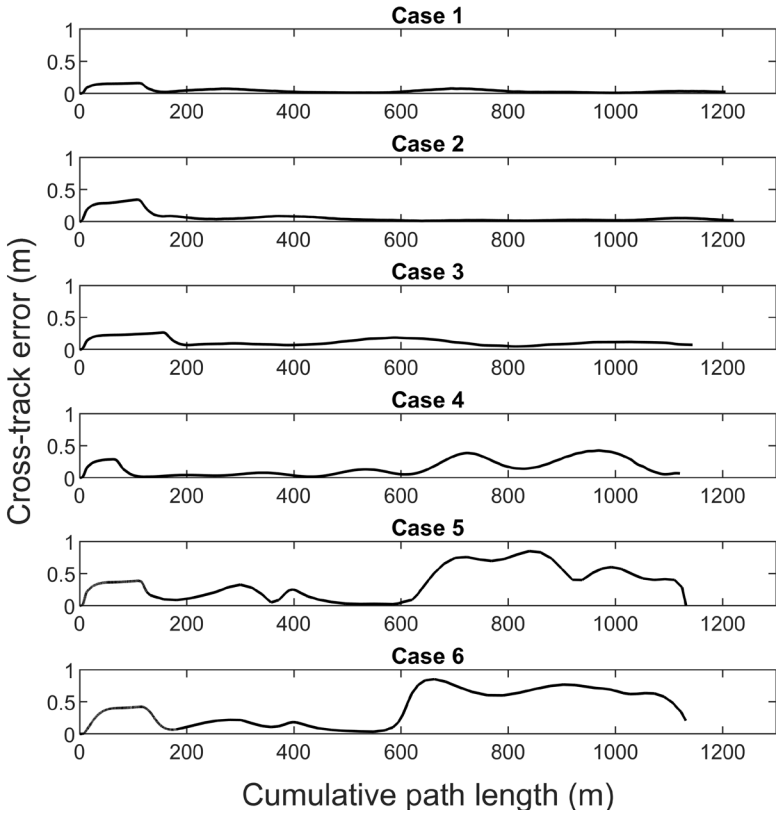


Figure 5.11: Cross-track errors between executed paths and planned paths.

Next, the performance of the path replanner was assessed and compared with two other path planners based on the following properties: solution qualities, stabilities, convergence behaviours, and computational requirements. In order to study these properties, the fitness of the obtained solutions and the runtime required to obtain the solutions were analysed. The fitness of a path solution is simply the travel time required by the AUV to arrive at the target by following the generated path. Therefore, a shorter travel time is the indicator of higher solution quality and hence, a stronger searching ability.

Shapiro-Wilk test with a significance level of 0.05 was used to examine the normality of the simulation results. The normality test revealed that the data was not normally distributed. Hence, medians and interquartile ranges were used as indicators for solution quality and stability. Figure 5.12 and Figure 5.13 show the boxplots of the simulation results. In the boxplots, the whisker indicates the range of data. The horizontal lines inside the boxes show the medians. The upper and lower quartiles are represented by the upper and lower ends of the box, which indicates the interquartile range. The lower ends of the whiskers in the boxplot of travel time identifies the best-known time obtained by the path planners in each case.

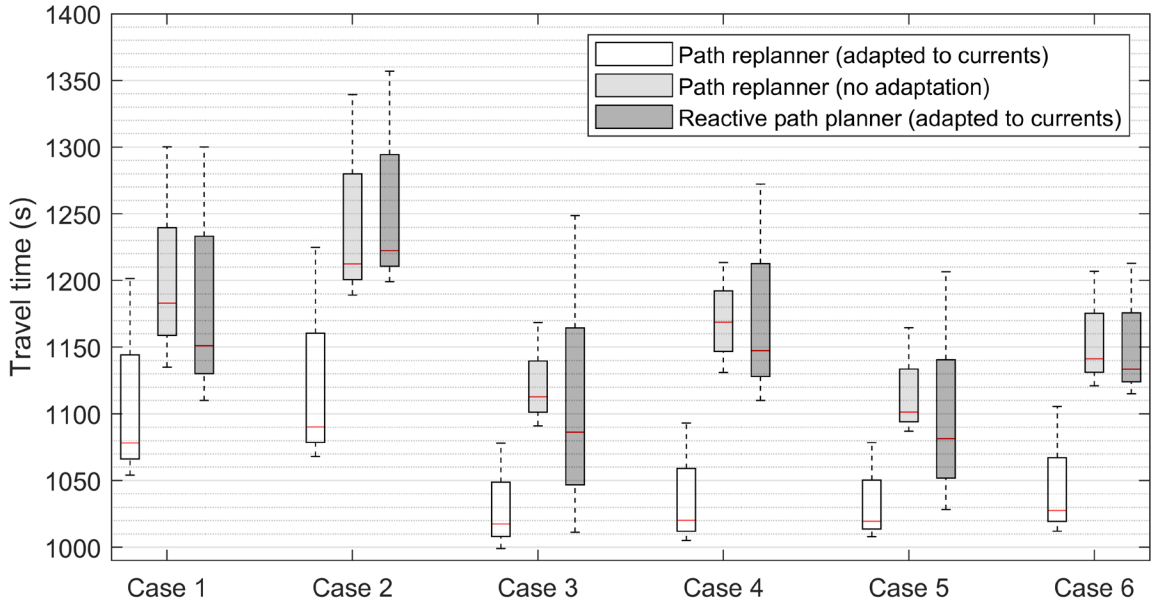


Figure 5.12: Travel time required by different path planners.

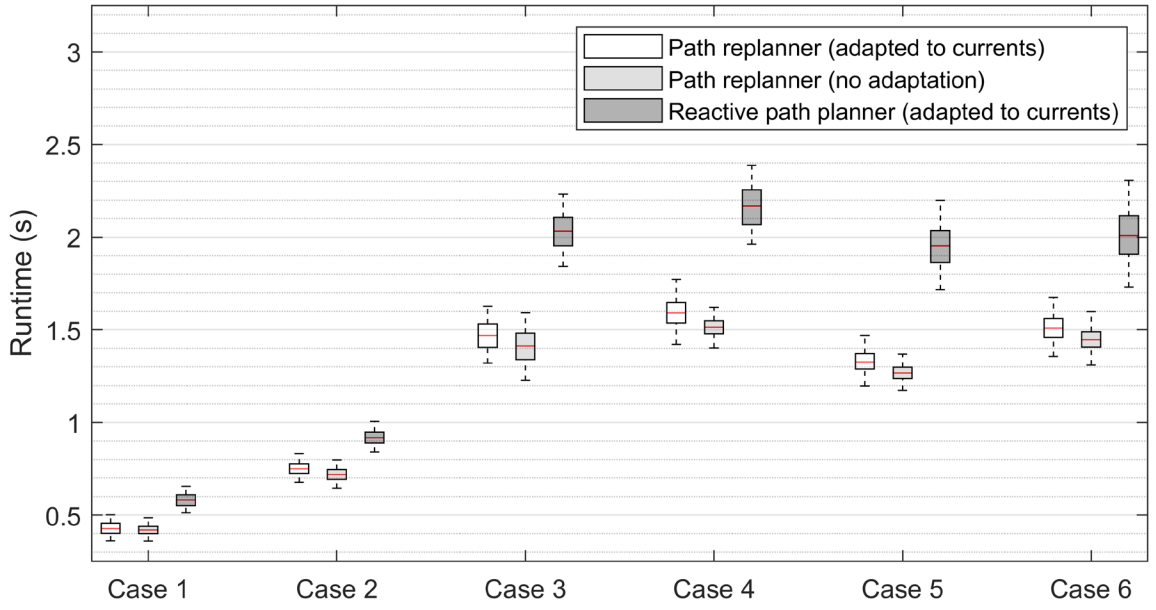


Figure 5.13: Runtime required by different path planners.

The effect of ocean currents on the AUV's performance was examined by comparing the proposed path replanner (adapted to currents) to the path replanner that was configured without the adaptation to currents. In contrast to the time-optimal path generated by the current-adapted path replanner, the second replanner simply searched for the path with the shortest distance. Figure 5.12 shows that the current-adapted path replanner generated

solutions with lower medians and best-known travel time in all test cases, suggesting a higher solution quality. The time-optimal path produced up to a 13% reduction in travel time compared to the path with the shortest distance. As the shortest-distance path did not take into consideration the effect of currents, the AUV that followed this path may run into adverse currents that opposed its motion and pushed it away from its path, leading to a less efficient operation. It was observed in Figure 5.13 that the additional computational load for adapting the path solutions to ocean currents caused a slight increase in the runtime required by the path replanner. The simulation results showed a maximum of 5% increase in runtime, which was found to be acceptable and insignificant (less than 80 ms).

When the path replanning scheme is compared with reactive planning, Figure 5.12 shows that the medians and the best-known travel time of the path replanner were better (lower) than the reactive planner in every test case. The path replanner was able to provide up to an 11% improvement in terms of travel time over the reactive planner, indicating the higher solution quality generated by the replanner. The path replanner was able to achieve better results because the replanning initialised the search for the optimal path from the previous solutions including the previously optimized path, whereas the reactive planner always initialised from the randomly generated solutions. This caused the reactive path planner to have poorer convergence and inadequate search before the iteration was stopped to output its final solution. In Figure 5.12, it can be observed in some cases that the best-known travel time obtained by the reactive path planner were close to the path replanner (less than 2% difference for Cases 3 and 5). This is because the reactive path planner also used the SDEQPSO algorithm, which is a metaheuristic optimization algorithm. This means that the stochastic solutions generated by the reactive planner also have the possibility to converge at the Pareto-optimal solution although it is less likely to occur. Nonetheless, the resultant medians of travel time produced by the reactive path planner were still worse than the replanner, leading to a significantly higher interquartile range in most test cases. The lower interquartile range of travel time generated by the path replanner indicates higher stability and robustness in all tested scenarios.

In terms of algorithm runtime, the path replanner also outperformed the reactive path planner as shown by the shorter average time required by the replanner in all test cases. The difference in their runtime is even more significant (up to approximately 30%) when the dimension of the problem increases to 3D. The reactive path planner required a longer

runtime because it needs to start afresh to search for the optimal path from the randomly generated solution every time, leading to a lower rate of convergence and inefficient computation. The path replanner has faster convergence and thus shorter runtime required as a result of reusing the previous solution to effectively search for the new optimal path. The higher solution quality and shorter runtime required by the SDEQPSO path replanner indicate its higher computational efficiency. Furthermore, the consistent performance of the path replanner throughout the Monte Carlo simulation under stochastic processes verifies its robustness in generating a safe and feasible AUV path.

5.5 Chapter Summary

In this chapter, the SDEQPSO algorithm has successfully employed for an online path replanner of an AUV in a dynamic operational environment. Using the onboard measurements from various sensor configurations, the proposed path replanner incorporated the effect of ocean currents in path optimization to continuously generate a time-optimal path for the AUV throughout its mission. Based on the simulation results, the time-optimal path generated by the proposed path replanner offered up to a 13% reduction in travel time compared to a path replanner that neglected the effect of currents. The proposed path replanning technique was also proven to have better performance over a reactive path planner in terms of solution quality (up to an 11% reduction in travel time), stability and computational efficiency (up to a 30% reduction in runtime). With the verified robustness through the Monte Carlo method, the generated path fulfilled the vehicle's safety and vehicular constraints, while taking into consideration the effect of ocean currents to improve the vehicle's operational efficiency. In summary, the proposed path replanner offers the following advantages:

- It generates time-optimal paths by using a computationally efficient algorithm to improve an AUV's performance.
- It is adaptive to the spatiotemporal variability of cluttered ocean environments and the constraints imposed by missions and vehicles.
- It does not require pre-generated paths, system models or any prior knowledge of the terrain/environment.
- It demonstrates its scalability for missions that require different setups of onboard sensors.

Chapter 6.

Implementation of an Online Path Replanner using MOOS-IvP

This chapter[5] describes the implementation of an online path replanner in actual AUVs by using an open-source system architecture, MOOS-IvP. The implementation was based on a modular framework, which was achieved by the operating system kernel that can run independent applications for each vehicular system module. This ensures the robustness of the planner during a mission. The performance of the path replanner was evaluated and verified under stochastic processes in hardware-in-the-loop (HIL) tests, in which the planner interacted with the onboard controllers and actuators of an Explorer AUV. The experimental results showed the path replanner was robust and capable of generating time-optimal paths to improve the AUV performance in an unexplored, cluttered and dynamic environment.

6.1 Real-time Implementation of Algorithms

Although extensive research has been conducted over the years to propose various path planning techniques for AUV missions, only a small number of them were verified experimentally or implemented in a real-time system. The majority of existing path planners, especially sophisticated online planners, were based on pure numerical simulations due to technical challenges and costs associated with implementation in an actual vehicle. Numerical studies often did not consider an AUV's actual capability to compute the paths and execute them in real time. It is important to assess the performance and robustness of a path planner when implemented in a real-time vehicle system. In particular, an online path planner must be examined for its capability in handling real-time feedback from the vehicle and its environments.

[5] This chapter was modified from the following publication: Lim, H. S., King, P., Chin, C. K. H., Chai, S., & Bose, N. (2021). Real-time implementation of an online path replanner for an AUV operating in a dynamic and unexplored environment (submitted and under the review of *Applied Ocean Research*).

Hardware-in-the-loop (HIL) test is a technique used for developing and testing real-time embedded systems and algorithms. A HIL test of an AUV can be conducted on dry land without requiring the deployment of the vehicle in water. It can serve as a platform for effective prototyping of AUV control algorithms in a safe and controlled environment before deploying the algorithms for actual field operations. Implementation of a path planner in a HIL test provides insight into how the planner interacts with the high-level and low-level controllers of an AUV in real time.

During the HIL test of an AUV, an interface is required to enable information exchange between the actual vehicle system and the algorithm under test. The Mission Oriented Operating Suite (MOOS) is a middleware that can serve as an interface for a HIL test. In the industry of marine robotics, the MOOS framework is commonly applied for field operations because of its wide-ranging capabilities and platform independence (Newman, 2008). Therefore, MOOS is often used for HIL tests as well as numerical simulations to allow for an easy transition into field operations. Some examples of existing MOOS implementations were presented by Paull et al. (2013), Hudson and Seto (2014), McMahon and Plaku (2016), Ferri et al. (2018), and Benjamin et al. (2019).

MOOS (Newman, 2008) is an open-source and cross-platform middleware that is developed to support robotic research. It adheres to the ISO (ANSI) C++ standard to ensure platform independence. Based on a publish-subscribe architecture, the MOOS middleware functions as a centralized database to enable information exchange between a community of vehicle system processes, which are run independently as MOOS applications. The MOOS library contains a collection of essential applications for robotic operations, ranging from autonomy, sensor management, and communication to debugging and data postprocessing. This reduces the time required to prototype a control algorithm when MOOS is used.

MOOS-IvP (Benjamin et al., 2010) is a comprehensive marine autonomy suite, in which the MOOS software is complemented by additional MOOS applications, including the IvP helm. The IvP helm uses interval programming (IvP) and a behaviour-based architecture to solve the multi-objective optimization of competing behaviours in a robotic system. The IvP helm represents each objective function as a behaviour that uses a piecewise linear approximation. As a single MOOS application, the IvP helm controls the desired speed, depth, and orientations of a vehicle by arbitrating multiple behaviours. The MOOS-IvP software was used to set up the HIL test in this chapter.

6.2 HIL Test Setup

The SDEQPSO path replanner proposed in Chapter 5 was implemented using the MOOS-IvP framework and analysed in a real-time HIL test. The MOOS-IvP framework is modular and uses a kernel that runs all MOOS applications independently to ensure the robustness of the system. The HIL test was set up using a backseat driver paradigm as described in Figure 6.1 and Table 6.1. The backseat driver separates the autonomy of an AUV from its vehicle control. The frontseat controller of the vehicle onboard computer executes the vehicle control, while the payload computer in the backseat is responsible for the vehicle autonomy.

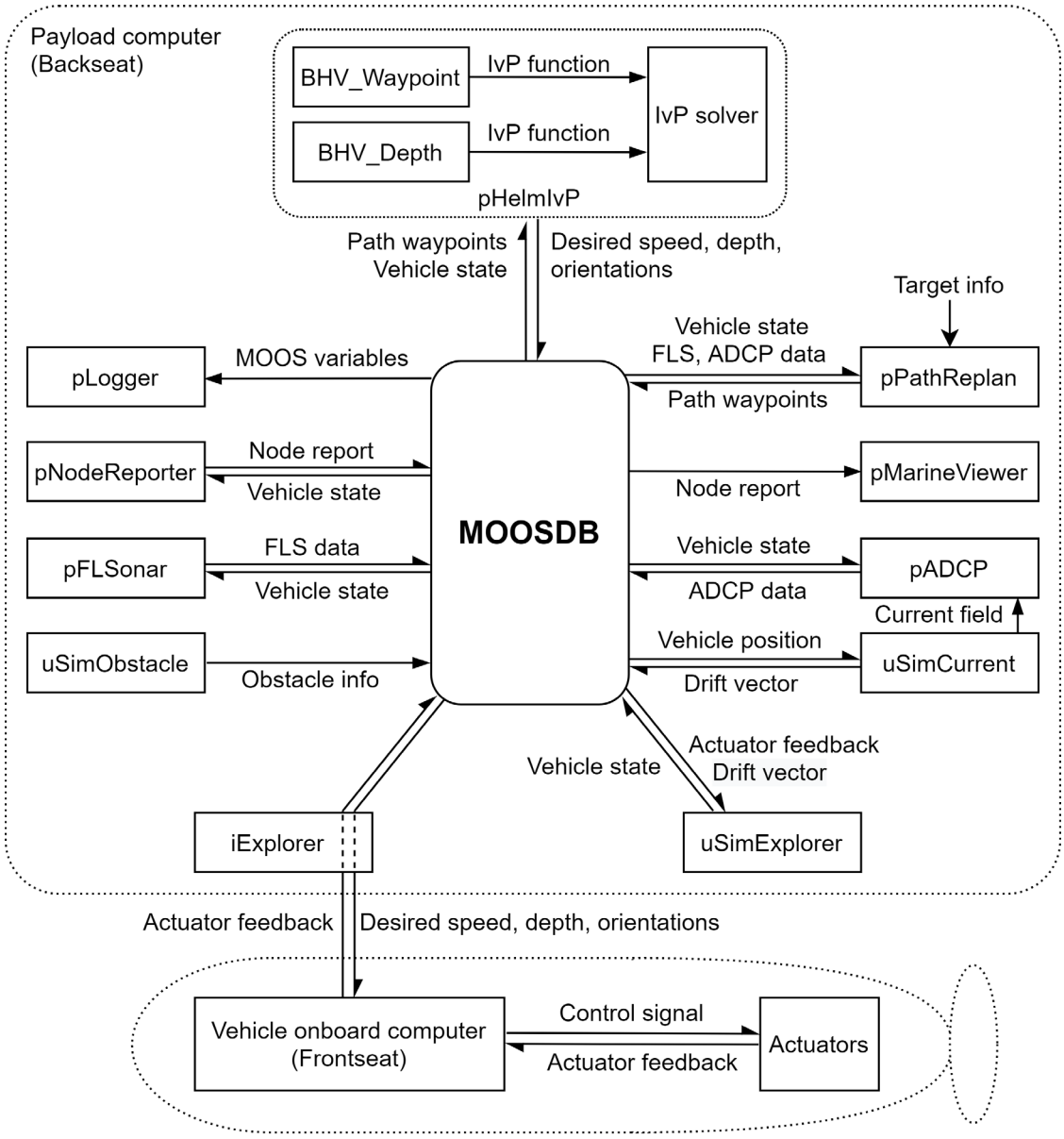


Figure 6.1: Backseat driver paradigm of HIL test using MOOS-IvP.

Table 6.1: Descriptions of components in the HIL test.

Component	**Type**	**Description**
MOOSDB	MOOS database	A centralized database for the communication between all MOOS applications.
pPathReplan	MOOS application	Generate a time-optimal path towards the target by using the SDEQPSO path replanner.
pNodeReporter	MOOS application	Generate a node report that consists of the vehicle info, including its position, speed, heading, etc.
pMarineViewer	MOOS application	Receive the node report to produce a visualization of vehicles and associated information.
pLogger	MOOS application	Record all variables sent between applications during a test for post-mission analysis, data gathering and post-mission replay.
uSimExplorer	MOOS application	A 3D vehicle simulator of an Explorer AUV.
iExplorer	MOOS application	Establish serial communications between the payload computer and the onboard computer.
uSimObstacle	MOOS application	Simulate static and dynamic obstacles that are *a priori* unknown to the vehicle.
pFLSonar	MOOS application	Simulate the data detected by an FLS sensor based on the vehicle position, orientations and obstacle information.
uSimCurrent	MOOS application	Simulate time-varying ocean currents that are *a priori* unknown to the vehicle.
pADCP	MOOS application	Simulate the data of a current field inferred by an H-ADCP sensor.
pHelmIvP	MOOS application	IvP helm for arbitrating multiple behaviours through an IvP solver.

Table 6.1 (continued).

Component	Type	Description
BHV_Waypoint	IvP behaviour	Path following behaviour for traversing a sequence of specified path waypoints.
BHV_Depth	IvP behaviour	Depth control behaviour for tracking a sequence of specified depths.
Onboard computer	Hardware	Generate control signals for shaft speed and rudder angles based on the desired speed, depth and orientations. Collect feedback from the actuators.
Actuators	Hardware	Actuate the propeller and control surfaces based on the control signals. Provide feedback on actual shaft speed and rudder angles.

During the test, the physical actuators of the AUV were controlled directly by the vehicle's embedded computer, while the hull remained stationary on dry land. The problem space was a virtual ocean environment, which was simulated by the uSimCurrent and uSimObstacle applications. The motion response of the AUV was modelled in the test by using the uSimExplorer application, which is a mathematical model of an Explorer AUV. The vehicle model is described in the next section.

6.2.1 Vehicle Model

In this chapter, the UTAS Explorer AUV "*nupiri muka*" was used as the test platform for verifying the proposed path planning system. The AUV is 7.5 metres in length, 0.74 metres in diameter and has a dry weight of 2000 kg. The actuators of the AUV include a two-bladed propeller, a set of X-form rudders, and a pair of hydroplanes. The AUV is equipped with an FLS sensor, two ADCP sensors, and a dual-frequency side-scan sonar. The modularity of the vehicle allows additional sensors to be configured as required by its missions. The control architecture of the vehicle is based on the MOOS-IvP framework. The vehicle is also equipped with a backseat driver system, which enables seamless information/data exchange between the vehicle and its operators by using the MOOS-IvP middleware. HIL tests of the vehicle were facilitated by the backseat driver. The detailed specification of the "*nupiri muka*" AUV can be found in Appendix B.

To conduct HIL tests on dry land, the AUV's motion response was simulated using an empirical model of the "*nupiri muka*" in the uSimExplorer application. The vehicle model can propagate the AUV's position and orientation based on the deflection angles of control surfaces and the propeller rotation rate, which were provided by the real-time actuator feedback from the AUV's hardware during the test. To simulate the effect of ocean currents, the vehicle model considered the drift force caused by time-varying currents by subscribing to the drift vector published by the uSimCurrent application. The vehicle model was governed by Equations (6.1) – (6.10).

$$v_{\dagger} = \min\left(v_{\max} \cdot \frac{rpm_{\dagger}}{rpm_{\max}},\ \dot{v}_{\max} \cdot d\dagger \right) \tag{6.1}$$

$$\Delta\Psi_{\dagger} = \min\left(\frac{-\alpha 3_{\dagger} + \alpha 4_{\dagger} - \alpha 5_{\dagger} + \alpha 6_{\dagger}}{4} \cdot \frac{\dot{\Psi}_{\max}}{\alpha_{\max}} \cdot \sqrt{\frac{rpm_{\dagger}}{rpm_{\max}}},\ \dot{\Psi}_{\max} \right) \tag{6.2}$$

$$\Psi_{\dagger} = \Psi_{\dagger\text{-}1} + \Delta\Psi_{\dagger} \tag{6.3}$$

$$\Delta\Theta_{\dagger} = \min\left(\frac{-\alpha 3_{\dagger} - \alpha 4_{\dagger} - \alpha 5_{\dagger} - \alpha 6_{\dagger}}{4} \cdot \frac{\dot{\Theta}_{\max}}{\alpha_{\max}} \cdot \sqrt{\frac{rpm_{\dagger}}{rpm_{\max}}},\ \dot{\Theta}_{\max} \right) \tag{6.4}$$

$$\Theta_{\dagger} = \Theta_{\dagger\text{-}1} + \Delta\Theta_{\dagger} \tag{6.5}$$

$$v_{hor} = \cos(\Theta_{\dagger}) \cdot v_{\dagger} \tag{6.6}$$

$$v_{ver} = -\sin(\Theta_{\dagger}) \cdot v_{\dagger} \tag{6.7}$$

$$x_{\dagger} = x_{\dagger\text{-}1} + \left[\sin(\Psi_{\dagger}) \cdot v_{hor} + drift\_x_{\dagger} \right] \cdot d\dagger \tag{6.8}$$

$$y_{\dagger} = y_{\dagger\text{-}1} + \left[\cos(\Psi_{\dagger}) \cdot v_{hor} + drift\_y_{\dagger} \right] \cdot d\dagger \tag{6.9}$$

$$depth_{\dagger} = \max\left[0,\ depth_{\dagger\text{-}1} + \left(v_{ver} - v_{buoy} \right) \cdot d\dagger \right] \tag{6.10}$$

where $\dagger$ is the current time, $d\dagger$ is the timestep and $v_{\dagger}$ is the vehicle speed, which can be resolved into a horizontal component v_{hor} and a vertical component v_{ver}. The vehicle position can be given by $x_{\dagger}$, $y_{\dagger}$, and $depth_{\dagger}$. The vehicle heading and pitch angle are denoted by $\psi_{\dagger}$ and $\theta_{\dagger}$, respectively. $rpm_{\dagger}$ is the propeller rotation rate, and $\alpha n_{\dagger}$ is the deflection angle of the n^{th} control surface. The drift velocity in the x and y direction are represented by $drift\_x_{\dagger}$ and $drift\_y_{\dagger}$.

The kinematic coefficients of the model are given in Table 6.2 and were based on the AUV's performance data, which were obtained from its field experiments (Pyper, 2018; Spain et al., 2019). The performance of the vehicle model was validated against the field data of the *nupiri muka* AUV in Appendix C.

Table 6.2: Kinematic parameters of the vehicle model.

Parameter		**Value**
$v_{\max}$	Maximum body-fixed velocity	2.2 m/s
$\dot{v}_{\max}$	Maximum rate of change of body-fixed velocity	0.1 m/s^2
$\dot{\psi}_{\max}$	Maximum rate of change of vehicle heading	5°/s
$\dot{\theta}_{\max}$	Maximum rate of change of vehicle pitch	3°/s
$rpm_{\max}$	Maximum achievable propeller rotation rate	300 rev/min
$\alpha_{\max}$	Maximum deflection angle of control surfaces	23°
v_{bouy}	Static upward velocity due to buoyancy	0.06 m/s

6.3 Experiments and Results

The implementation of the SDEQPSO path replanner using the MOOS-IvP framework was evaluated by performing a HIL test on the "*nupiri muka*" AUV. The payload computer was a Linux machine with Ubuntu 18.04 (GNU g++ 7.5.0) and Intel Core i5-6300U (2.4GHz CPU, 8GB RAM). The problem space was virtual ocean environments with 1000×1000 square metres for 2D and 1000×1000×50 cubic metres for 3D. The target was located at (780, 780) for 2D and (780, 780, 15) for 3D, which require an r_{target} of 1103.1 metres and 1103.2 metres, respectively. *A priori* unknown obstacles and time-varying ocean currents were simulated in the problem space. The virtual obstacles were placed in such a way that they will potentially block the AUV paths. Dynamic obstacles were configured to move independently in different directions at random speeds up to 0.1 m/s. Time-varying current field with current velocity up to 0.5 m/s was generated based on field data of ocean currents. The field data were obtained at Beauty Point, Tasmania, Australia by using the ADCP sensors of the "*nupiri muka*" AUV.

The population size of the SDEQPSO algorithm was set to 150 particles. The algorithm parameters were configured based on the suggestions in Chapter 3. Different test cases as described in Table 6.3 were conducted during the HIL test. Based on the properties of the "*nupiri muka*" AUV, the safety distance and buffer distance required for obstacle avoidance were defined as 10 metres and 30 metres, respectively. The AUV was configured to travel with a reference speed of 1.5 m/s. When the horizontal FLS configuration was used, the azimuthal boundary $\Phi_{\max}$ was set to ±60° because the FLS model had a field of view of 120°. For the vertical FLS configuration, the polar boundary $\Theta_{\max}$ was set to ±20° as the pitch angle of *nupiri muka* rarely exceeds 20° during operations.

The compatibility of these settings was checked against the kinematic properties of *nupiri muka*. The vehicle has a maximum yaw rate of 5°/s and a maximum pitch rate of 3°/s. With a preset reference speed of 1.5m/s, this corresponded to a heading change of 3.33° and a pitch change of 2° per metre. Due to the radial constraint, the minimum length of two consecutive path segments must be greater than the radial distance r_d, which was set as 50 metres. This resulted in an allowable heading change of 166.5°, providing sufficient yaw motion for the range of azimuthal angle Φ (120°). This also allowed for a pitch change of 100°, which was sufficient for the range of polar angle Θ (40°). Thus, these settings were compatible with the *nupiri muka* AUV.

Table 6.3: Setups of HIL test cases.

Test Case	**Dimension**	**FLS configuration**	**Obstacles**	r_d **(m)**	Φ_{max} **(°)**	$\Theta_{\max}$ **(°)**
1 (surface test)	2D	Horizontal	Stationary	50	±60	-
2 (surface test)	2D	Horizontal	Moving	50	±60	-
3 (dive test)	3D	Horizontal	Stationary	50	±60	±5
4 (dive test)	3D	Horizontal	Moving	50	±60	±5
5 (dive test)	3D	Vertical	Stationary	50	±5	±20
6 (dive test)	3D	Vertical	Moving	50	±5	±20

The robustness of the SDEQPSO path replanner was assessed under scenarios with stochastic processes, *i.e.*, randomly moving obstacles and constantly changing ocean currents. To adapt an AUV path to ocean currents, the path replanner can accept current profiles in the form of real-time ADCP data or a predictive ocean model. A comparison was conducted between the path replanners that used different current data:

1. Real-time current profile inferred from the H-ADCP measurements,
2. Pre-generative current profile loaded directly from the ocean current data.

The path replanner was also evaluated by comparison with two other planners:

1. An SDEQPSO-based path replanner without adaptation to ocean currents,
2. A local path planner that employed the MOOS-IvP inbuilt dynamic obstacle manager (Benjamin, 2020).

The HIL test of the "*nupiri muka*" was conducted under the scenario described in Figure 6.2 and Figure 6.3. The mission of the AUV was to traverse the unknown, cluttered, and dynamic ocean field towards a target while maintaining a safe distance with the virtual obstacles and attempting to exploit the favourable currents that can assist the vehicle motion. During the surface tests (Cases 1 and 2), the target and the obstacles were placed on the ocean surface. For the dive tests (Cases 3, 4, 5, and 6), the target was located at a depth of 15 metres, while the obstacles were placed at various depths between the surface and the target depth.

Figure 6.2: Bow view (left) and stern view (right) of the "*nupiri muka*" AUV during the HIL tests.

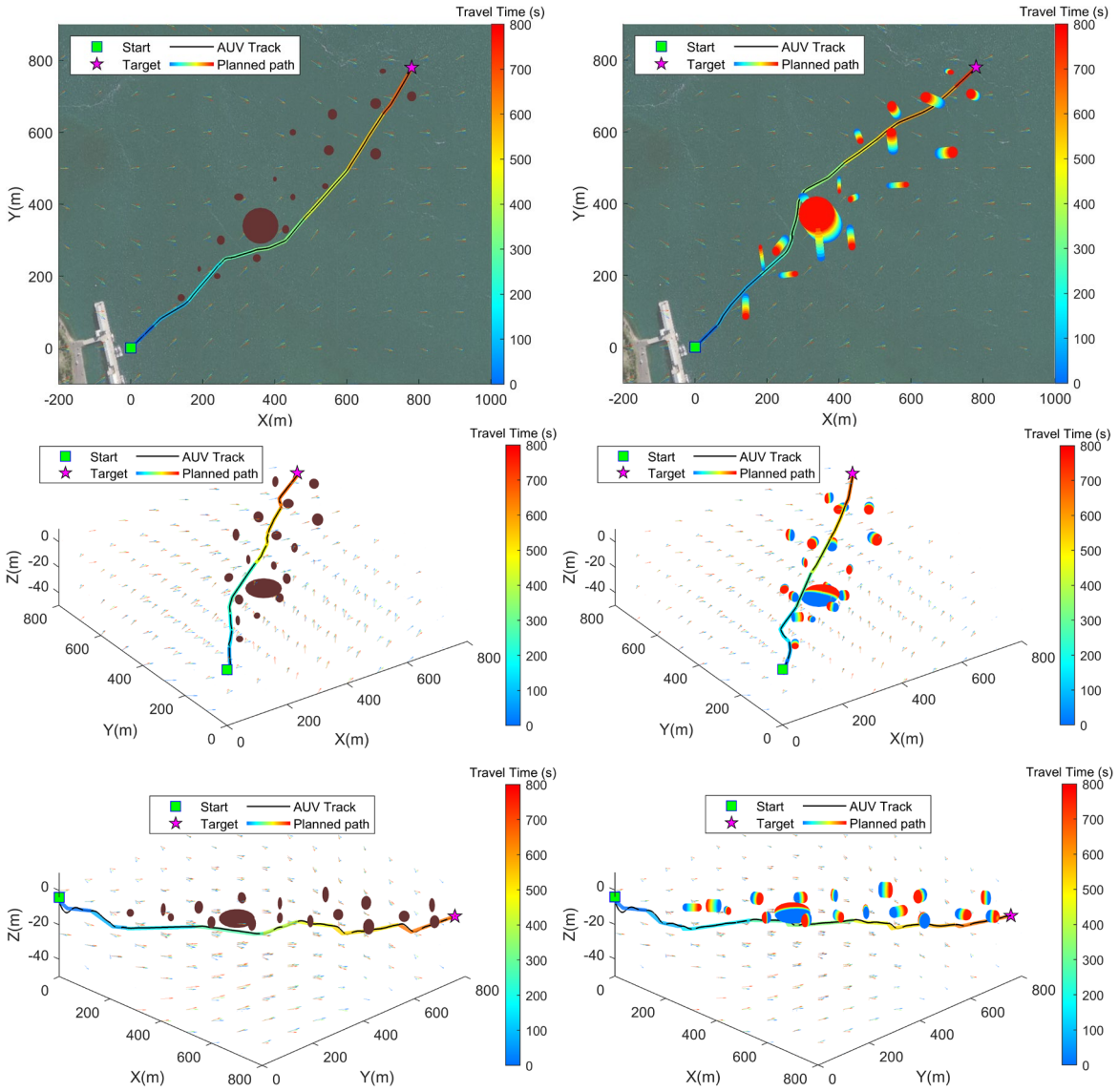


Figure 6.3: Path solutions of Case 1 (top-left), Case 2 (top-right), Case 3 (mid-left), Case 4 (mid-right), Case 5 (bottom-left), and Case 6 (bottom-right).

In Figure 6.3, the elapsed time during the tests is shown by the colour bars. The planned paths and vector fields of ocean currents are coloured according to the elapsed time. The static obstacles (Case 1, 3 and 5) are coloured brown, while the moving obstacles (Case 2, 4 and 6) are represented by the trails with colours corresponding to the elapsed time. A path is safe if it does not intersect with the brown static obstacles or the trails of moving obstacles with the same colour. The virtual obstacles were configured to obstruct the vehicle path intentionally so the AUV must use its sensors to detect and detour around the obstacles by replanning its path. When the vehicle used a horizontal sonar configuration (Case 3 and 4), the path replanner required it to manoeuvre around obstacles mainly by using yaw motion.

Conversely, the planned path for the vehicle that used a vertical sonar configuration (Case 5 and 6) involved mostly pitch motion.

Throughout the experiment, the AUV was required to use its onboard controller and physically actuate its control surfaces and propeller in order to follow the planned path. In all test cases, the AUV maintained an average vehicle speed of 1.5 m/s. Figure 6.3 shows that the resultant paths are safe and collision-free. The executed vehicle tracks (black lines) showed close resemblances to the planned paths, indicating that the planned paths can be followed closely by the AUV in real time. By following a time-optimal path, the vehicle was driven to surf the favourable currents and to keep away from the adverse currents that would oppose its motion.

The path replanner must replan the AUV path in real time whenever its replanning flags were triggered, *e.g.*, the previously planned paths conflicted with obstacles detected by the FLS sensor. Figure 6.4 shows the variation of runtime for the path replanner to replan the vehicle path during the real-time missions when it was implemented in the MOOS-IvP framework. In the boxplot, the lower and upper ends of the boxes indicate the first and third quartiles of the runtime respectively, while the red lines across the boxes show the medians. During the HIL test, the runtime required for path replanning varied from milliseconds up to a maximum of 2.7 seconds (Case 3). The means of runtime in all test cases were found to be below 1 second. The resulting runtime shows that the path replanner can be run smoothly on the payload computer or an embedded system that uses the MOOS-IvP middleware.

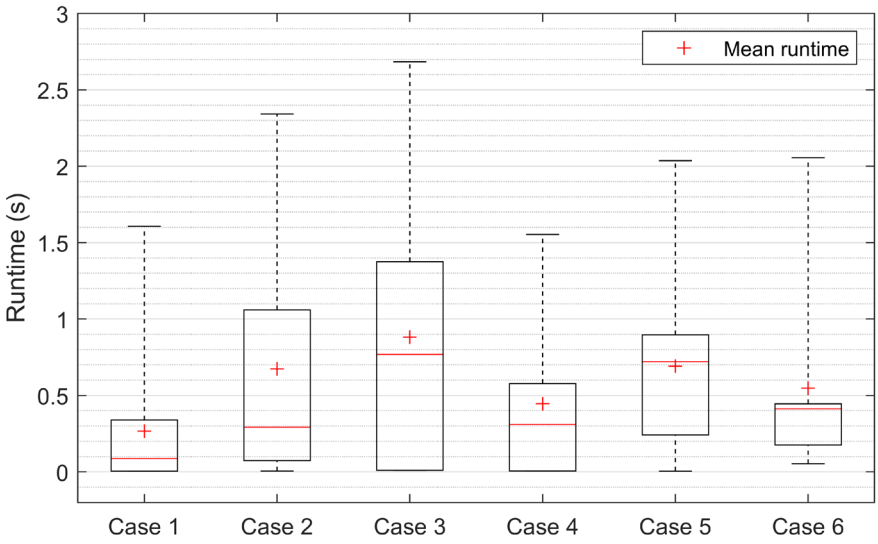


Figure 6.4: Runtime of the implemented SDEQPSO path replanner for replanning paths in different test cases.

The feasibility of the planned paths was checked against the physical limitations of the AUV actuators. The "*nupiri muka*" AUV has a minimum turning radius of 10 metres and a maximum pitch of 40° in the worst-case scenario. Figure 6.5 shows that the curvature radius of the planned paths was maintained higher than the minimum turning radius of the vehicle in the test cases.

The vehicle heading and pitch of the *nupiri muka* AUV when following the planned paths are depicted in Figure 6.6 and Figure 6.7, respectively. The variation of vehicle pitch in indicates that the required actuation was well within the vehicular limitation. Based on Figure 6.6 and Figure 6.7, when the horizontal sonar configuration was used (Cases 1, 2, 3 and 4), the planned paths required the vehicle to manoeuvre mainly by using yaw motion. On the other hand, when the vertical sonar configuration was used (Cases 5 and 6), the paths required mostly pitch motion. The generated control signal for the AUV to follow the planned paths in all cases can be found in Appendix E.

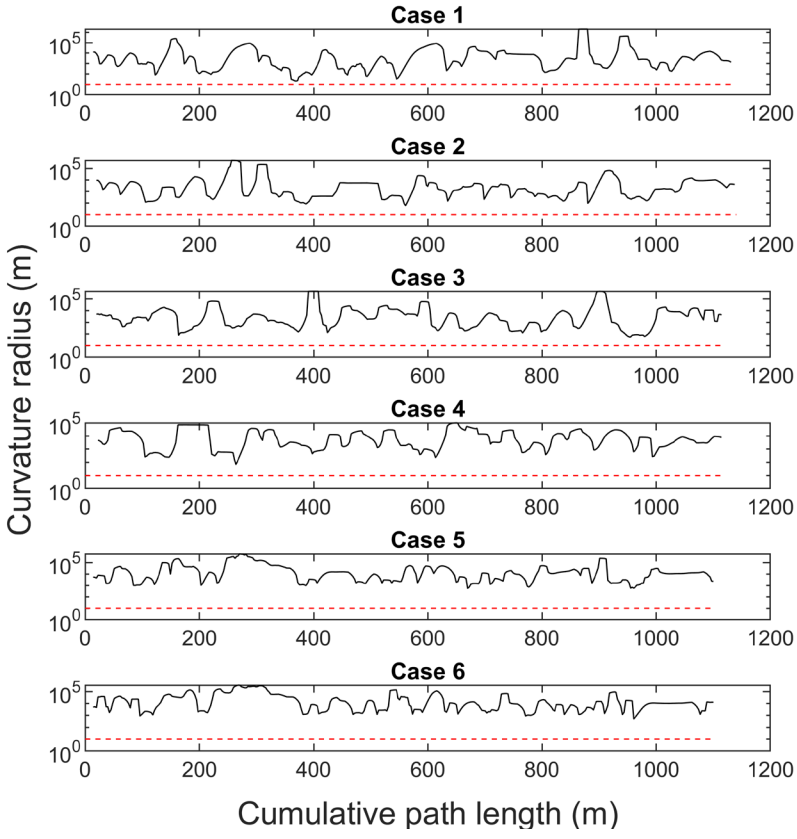


Figure 6.5: Variation of path curvature radius with respect to physical limitations.

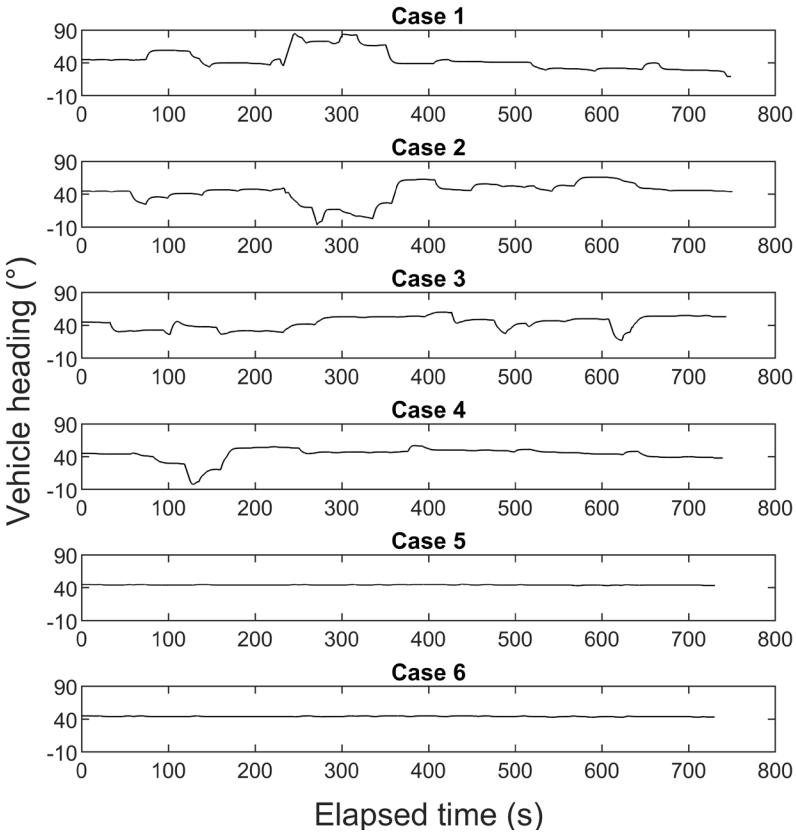


Figure 6.6: Variation of vehicle heading.

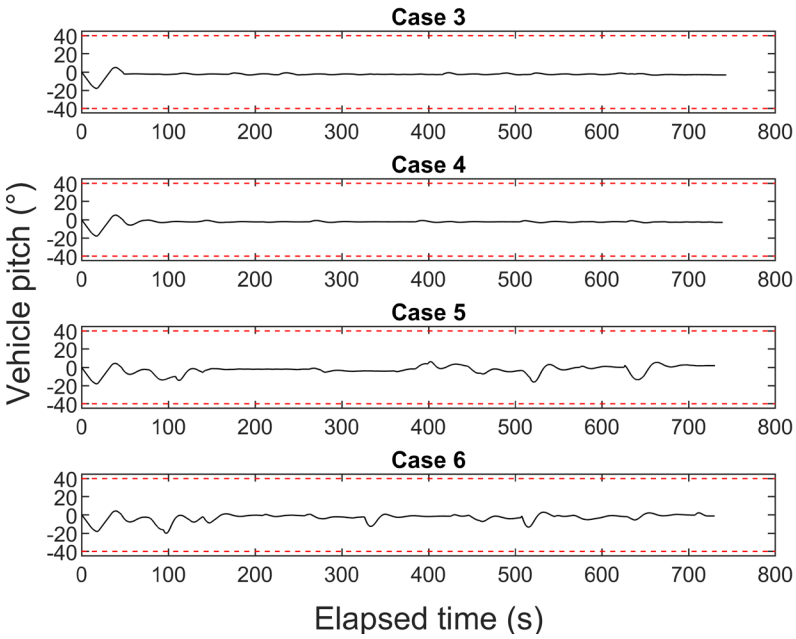


Figure 6.7: Variation of vehicle pitch with respect to physical limitations.

To examine the path following performance of the AUV, Figure 6.8 depicts the resultant cross-track error between the planned path and the actual path executed by the vehicle. Based on Figure 6.8, the variation of cross-track errors can be correlated with the occurrence

of path replanning. In the test cases, increases in the cross-track errors were observed mainly when a path change was required after the AUV received a replanned path. In some cases, multiple triggers of path replanning can be observed within a short distance. This was because the FLS sensor detected new obstacles that were either conflicting with the planned path or within the safe zone of the vehicle when the vehicle changed its orientation.

The cross-track errors were found to be less than 0.2% of the total length of the path travelled by the vehicle. This corresponds to a maximum error of 2 metres, which is deemed acceptable given the total distance travelled and the size of the Explorer AUV (7.5 metres long). The safety distance (10 metres) and buffer distance (30 metres) used by the path replanner were able to tolerate the resultant cross-track error. The results showed that the paths generated by the path replanner can be safely followed by the vehicle under the influence of drift caused by ocean currents. Thus, the practicability of the path replanner to generate feasible AUV paths in real time was verified.

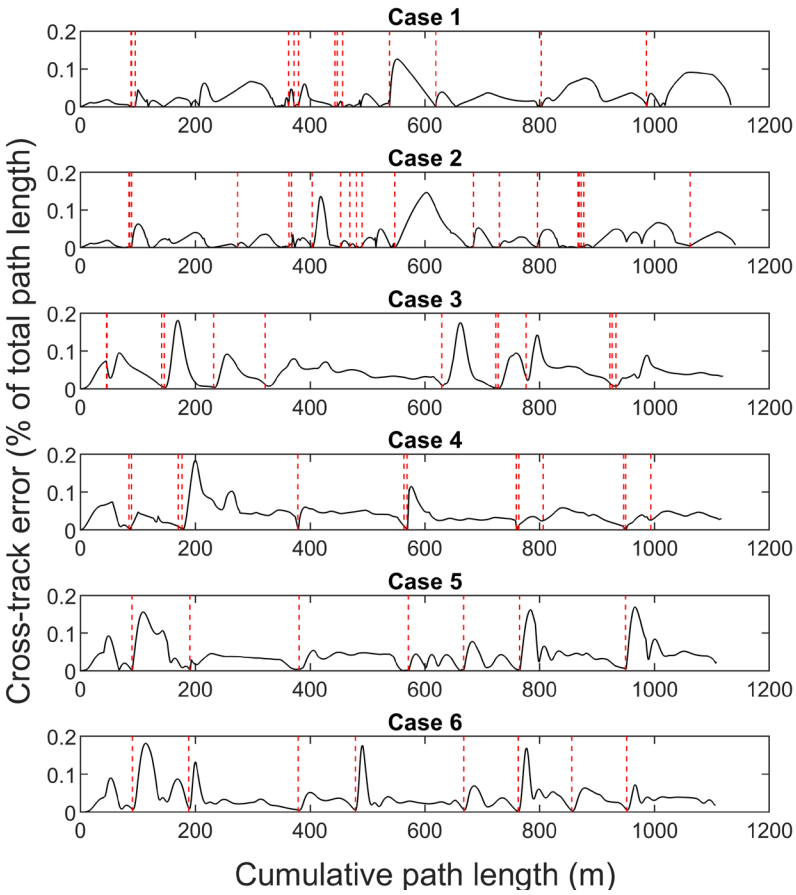


Figure 6.8: Variation of cross-track error between executed paths and planned paths (red dashed lines indicate when the vehicle received a replanned path).

To evaluate the performance of the proposed path replanner, the travel time required by the AUV to arrive at the target by following the paths generated by different path planners is collated in Figure 6.9. The local path planner was configured to follow a straight-line path that linked the starting point and the target. During the missions, the local path planner made local deviations to the planned path to avoid detected obstacles as required, and it returned the path after avoiding the obstacles. Cases 5 and 6 for the local path planner were excluded because the planner is only capable of making local deviations by changing the vehicle's heading, and thus, it is not compatible with the vertical sonar configuration.

The results in Figure 6.9 indicate that the proposed path replanner produced a shorter travel time than the local path planner. The path replanner reduced the AUV's travel time by 4 – 16% in 2D scenarios (Cases 1 and 2) and by 3 – 12% in 3D scenarios (Cases 3 and 4). The effectiveness of the path replanner in reducing the travel time varies depending on the type of current profile data used. The local path planner resulted in a longer travel time because it did not consider the optimality of the path; instead, it simply followed a straight-line path and deviated from the path to avoid obstacles. After passing the obstacles, instead of continuing to head towards the target, the local path planner always caused the vehicle to steer back to the original path regardless of its position. Furthermore, the solutions of the local path planner cannot be adapted to ocean currents.

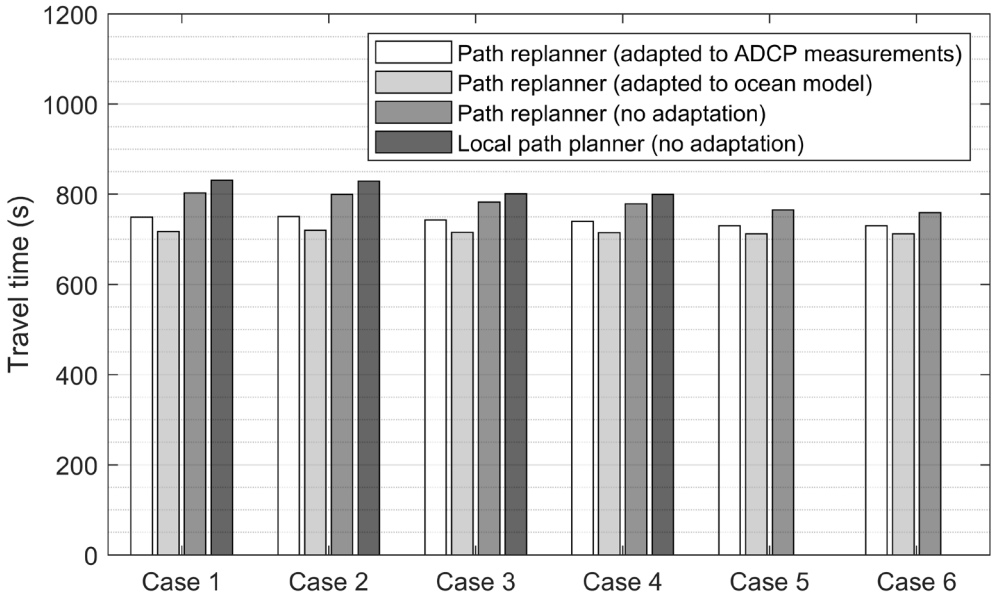


Figure 6.9: Travel time obtained by different path planners.

To examine the effect of ocean currents on the AUV's performance, the path replanner was configured to use different types of current profile data for adapting its solutions to currents. Contrary to the current-adapted path replanner that generated time-optimal paths, when the path replanner was configured without the adaptation to ocean currents, it simply searched for the path with the shortest distance towards the target. As shown in Figure 6.9, the resultant travel time of the two current-adapted path replanners was shorter in all test cases when compared to the path replanner that was not adapted to currents. By travelling on time-optimal paths, the AUV's travel time was reduced by up to 7% when the ADCP measurements were used, and up to 12% when the predictive ocean model was used. Travelling on the shortest-distance path, which did not consider the effect of currents, required a longer travel time to reach the target because the vehicle did not attempt to exploit favourable currents or to avoid adverse currents that opposed its motion and pushed it away from its path.

Between the two current-adapted path replanner, the predictive ocean model resulted in better solutions than the real-time ADCP measurements in all test cases. Figure 6.9 shows a maximum of 5% reduction in travel time was achieved by the path replanner that used an ocean model. Although the ADCP sensor measures ocean currents in real time, its profiling range is limited, thus leading to less current profile information available for path planning. Despite not having a real-time accuracy, a predictive ocean model can further improve the effectiveness of time-optimal paths because it provides a more comprehensive current profile for path planning. This enables the path replanner to further exploit ocean currents for every component of its time-optimal path. However, predictive ocean models are not always available, especially when planning an AUV path in unknown underwater environments. The proposed path replanner provides versatility in accepting both types of current profile data to generate a time-optimal path.

6.4 Chapter Summary

In this chapter, the SDEQPSO path replanner was implemented as an independent application module in an AUV system that used the MOOS-IvP architecture. The implemented path replanner was verified in a HIL test of an Explorer AUV to analyse its interaction with the onboard controller and physical actuators. A variety of sensor configurations and test scenarios were involved in the experiment, which required the AUV to traverse across an unknown, dynamic and cluttered ocean environment. The proposed

path replanner demonstrated its scalability for missions that require different sensor configurations and its versatility to accept different current profile data. During the experiment, the path replanner seamlessly worked in conjunction with the hardware on the test platform in real time to continuously generate a time-optimal path that exploited ocean currents and maintained obstacle avoidance. The experimental results verified that the planned path can be followed closely by the AUV under the disturbance of ocean currents. By comparison with a local path planner, the path replanner improved the vehicle performance by reducing up to 16% of its travel time required to reach a target. When compared to the shortest-distance paths, the generated time-optimal path reduced the travel time by up to 7% when onboard sensor measurements were used, and up to 12% when a predictive ocean model was used.

By reducing an AUV's travel time and energy usage for transiting between survey locations, the path replanner can enable the vehicle to preserve its energy for conducting surveys. This reduces the necessity of retrieving and redeploying an AUV for recharging its battery during a long-duration mission, hence improving the vehicle's competence and mission efficiency.

This page is intentionally left blank.

Chapter 7.

Closing Remarks

This chapter concludes the thesis with a summary of the presented work. It highlights the key outcomes and proposes several directions for future work.

7.1 Summary and Conclusions

This thesis focuses on developing an online path planner to improve the operational performance of an AUV that operates in dynamic ocean environments. The study covered the algorithm development for various path planning and replanning scenarios and missions that have different requirements.

Following the literature review in Chapter 1, the research began with a preliminary review study in Chapter 2, which comprehensively compared the performance of existing PSO-based algorithms for solving the AUV path planning problem. The variants of PSO and QPSO were studied and classified into several categories based on the approaches used to improve the algorithm performance. A pre-generative AUV path planner was developed to solve the offline path planning problem of an AUV operating in a turbulent and cluttered ocean environment that was *a priori* known. The path planner aimed to generate safe and dynamically feasible time-optimal paths that could exploit ocean currents to improve the AUV's performance. Using Monte Carlo simulations in 2D and 3D scenarios, the algorithms were benchmarked based on their solution qualities, stabilities, convergence behaviours and computational requirements. The review successfully identified the strengths and weaknesses of different PSO-based algorithms with the following key findings:

- The hybridization of DE operation in PSO and QPSO provided up to a 9% improvement in the searching ability of particles to find the optimal path.

- The DE-hybridized algorithms have high computational requirements due to the greedy selection operator, which requires the particles to undergo fitness evaluation twice during every iteration.
- APSO and IPSO-SQP can achieve a balance between solution qualities and computational requirements, with their solution qualities slightly lower than the DE-hybridized algorithms.

Based on the preliminary review, Chapter 3 proposed a novel approach to improve the performance of PSO-based algorithms in solving an AUV path planning problem by using selective hybridization of DE. The proposed algorithms carry out the DE operation selectively on a number of particles to enhance the swarm's searching ability and resistance to local minima without inflating the computational cost. An empirical study and a benchmark study based on several non-linear continuous test functions were conducted to analyse the algorithms. The proposed algorithms were benchmarked against other algorithms in offline AUV path planning scenarios, which consist of *a priori* known obstacles of different sizes and non-uniform ocean currents. Based on the Monte Carlo simulations and Kruskal-Wallis ANOVA tests, the following outcomes were obtained:

- The proposed algorithms maintain a similar time complexity and spatial complexity to the standard PSO and QPSO algorithms based on the O notation.
- For the majority of the tested problems, the algorithms showed the optimal performance when using a selection factor S of 0.1 – 0.3, *i.e.*, 10 – 30% of particles were DE-hybridized. For path planning problems, the setting of $S = 0.3$ was found to be appropriate and effective.
- The proposed algorithms achieved similar performance to DEPSO and DEQPSO in terms of solution quality and stability, while having a significantly lower computational requirement (up to a 50% reduction in algorithm runtime).
- The performance of the proposed algorithms surpassed most of the tested algorithms (up to a 10% improvement in solution qualities), with the SDEQPSO algorithm ranked higher than the SDEAPSO algorithm.
- The time-optimal paths generated by the proposed algorithms were dynamically feasible for the REMUS 100 AUV.

Next, Chapter 4 formulated the objective function of the path planner as constrained optimization to improve its search efficiency, which is the most important attribute for an online path planner. The use of the polar coordinate system and a combination of hard and soft constraints allowed the vehicular constraints to be satisfied and facilitated the placement of path nodes to enhance the search efficiency of the path planner. Using Monte Carlo simulations and Kruskal-Wallis tests, different types of constraints were analysed to identify the optimal constraint setting that improved the search efficiency of the path planner. The key findings are as follows:

- The SDEQPSO path planner with hard-constrained boundary conditions and soft-constrained obstacle avoidance produced the highest search efficiency.
- The hard constraint setting for obstacle avoidance guaranteed the feasibility of a path but required significantly higher computational cost.
- The use of a hard constraint is inappropriate for the SDEAPSO algorithm due to impractically long initialisation runtime.
- The REMUS 100 simulation model can follow the time-optimal path generated by the constrained path planner.

Chapter 5 proposed an online AUV path planner that employed the SDEQPSO algorithm and a path replanning approach to optimize an AUV mission in an unknown, dynamic and cluttered ocean environment. Without requiring any system model or prior knowledge of the environment, the SDEQPSO path replanner incorporated the effect of ocean currents in path optimization to generate a time-optimal path based on the measurements from the H-ADCP and FLS sensors with different configurations. In addition to the verified robustness through the Monte Carlo method, the proposed path replanner achieved the following outcomes:

- It can continuously refine a safe and feasible path for an unknown environment based on the feedback from the REMUS 100 AUV model and its onboard sensors.
- The continuously refined paths can be followed by the REMUS 100 AUV under the disturbance of ocean currents in simulations.
- The time-optimal paths reduced the AUV travel time by up to 13% compared to the shortest-distance paths.

- The proposed path replanning scheme provided up to an 11% reduction in travel time and up to a 30% reduction in algorithm runtime compared to the reactive planning scheme.

In Chapter 6, the SDEQPSO path replanner was implemented as an independent application module in an open-source system architecture, MOOS-IvP. The modular implementation ensures the stability and robustness of the path replanner in a vehicle system. The implemented path replanner was verified in a HIL test of an Explorer AUV by analysing its interaction with the onboard controller and physical actuators. Based on the experiments that involved different sensor configurations and test scenarios, the following conclusions can be drawn about the path replanner:

- It can be run seamlessly with the hardware onboard an Explorer AUV in real time to continuously generate a safe and feasible path for unexplored operational environments.
- The continuously refined paths can be followed by the Explorer AUV in real time under the disturbance of ocean currents.
- It offered up to a 16% reduction in the AUV travel time compared to a local path planner that used a dynamic obstacle manager to follow a pre-planned path.
- The generated time-optimal path reduced the travel time by up to 7% when onboard sensor measurements were used, and further up to 12% when a predictive ocean model was used.

The resultant path replanner developed in this thesis contributes to enabling an AUV to achieve a higher level of autonomy and hence improve its competence in missions with longer durations. It can offer the following advantages:

- By using a computationally efficient algorithm, it generates time-optimal paths that exploit ocean currents to improve an AUV's performance.
- It continuously adapts the generated path to the spatiotemporal variabilities of operational environments, and the constraints imposed by missions and vehicles.
- It does not require pre-planned paths, system models or any prior knowledge of the terrain/environment.
- It demonstrated its scalability for AUVs of different sizes and with different actuators.

- It is scalable for missions that require different setups of onboard sensors.
- It has the versatility to accept different current profile data for generating time-optimal paths.
- Its implementation is based on a modular framework in an open-source system, which promotes ease of applications and extendibility for different robotic platforms.

7.2 Suggestions for Future Work

In this thesis, the proposed path replanner was tested in fully controlled environments. Extra care must be taken when deploying the path replanner in field operations. The presented work can be extended in several directions. The future work for developing the proposed path replanner is recommended as follows:

- The obvious next step for verifying the path replanner is to conduct in-water field experiments. The performance of the planner should be tested using actual sensor feedback because the sensor measurements used in this thesis were assumed to be reliable and noise-free. The proposed path replanner promotes the ease of transition into field operations.
- The proposed path replanner only supports the planning of a single AUV. It can be extended for applications in multi-vehicle cooperative operations. Using the MOOS-IvP middleware, it is possible to run the planner on multiple AUVs to facilitate their coordination during a mission, such as a rendezvous at a designated point.
- The path replanner does not incorporate risk awareness. It is possible to improve vehicle safety in a mission by introducing situational awareness and fault tolerance in the planner. By incorporating awareness for internal faults (such as hardware or software failures) and external faults (such as a hazardous or trapped environment), the planner can abort the path to its current target and plan a path for emergency surfacing if the severity of fault is not tolerable.
- The path replanner can be applied to marine vehicle systems other than an AUV, if the vehicle is equipped with a current profiler or when a predictive ocean model is available. The modularity of the implemented planner ensures its scalability and extendibility for different vehicle systems.

This page is intentionally left blank.

References

Ackley, D. H. (1987). The model. In *A Connectionist Machine for Genetic Hillclimbing* (pp. 29-70). Springer.

Akbarimajd, A. (2014). Optimal motion planning of juggling by 3-DOF manipulators using adaptive PSO algorithm. *Robotica*, *32*(6), 967-984.

Allen, B., Vorus, W. S., & Prestero, T. (2000). Propulsion system performance enhancements on REMUS AUVs. OCEANS 2000 MTS/IEEE, Providence, RI.

Alvarez, A., Caiti, A., & Onken, R. (2004). Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, *29*(2), 418–429.

Ambrosino, G., Ariola, M., Ciniglio, U., Corraro, F., De Lellis, E., & Pironti, A. (2009). Path generation and tracking in 3-D for UAVs. *IEEE Transactions on Control Systems Technology*, *17*(4), 980-988.

Bäck, T., & Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, *1*(1), 1-23.

Bakdi, A., Hentout, A., Boutami, H., Maoudj, A., Hachour, O., & Bouzouia, B. (2017). Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. *Robotics and Autonomous Systems*, *89*, 95-109.

Bakolas, E., & Tsiotras, P. (2013). Optimal synthesis of the Zermelo–Markov–Dubins problem in a constant drift field. *Journal of Optimization Theory and Applications*, *156*(2), 469-492.

Barisic, M., Miskovic, N., Vukic, Z., & Vasilijevic, A. (2010). Geometric Primitives-Based AUV Path Planning In Cluttered Waterspaces. *IFAC Proceedings Volumes*, *43*(20), 22-27.

Belkhouche, F. (2009). Reactive path planning in a dynamic environment. *IEEE Transactions on Robotics*, *25*(4), 902-911.

Belkhouche, F., & Bendjilali, B. (2012). Reactive path planning for 3-D autonomous vehicles. *IEEE Transactions on Control Systems Technology*, *20*(1), 249-256.

Benjamin, M. R. (2020). *pObstacleMgr: Managing Vehicle Belief State of Obstacles*. Massachusetts Institute of Technology. https://oceanai.mit.edu/ivpman/pdfs/app_pobstaclemgr.pdf

Benjamin, M. R., Defilippo, M., Robinette, P., & Novitzky, M. (2019). Obstacle avoidance using multiobjective optimization and a dynamic obstacle manager. *IEEE Journal of Oceanic Engineering*, *44*(2), 331-342.

Benjamin, M. R., Schmidt, H., Newman, P. M., & Leonard, J. J. (2010). Nested autonomy for unmanned marine vehicles with MOOS-IvP. *Journal of Field Robotics*, *27*(6), 834-875.

Biswas, S., Anavatti, S. G., Garratt, M. A., & Pratama, M. (2016). Simultaneous replanning with vectorized particle swarm optimization algorithm. 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), Phuket, Thailand.

Brock, O., & Khatib, O. (2000). Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths. IEEE International Conference on Robotics and Automation, San Francisco, CA.

Bruce, J., & Veloso, M. (2002). Real-time randomized path planning for robot navigation. IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland.

Caharija, W., Pettersen, K. Y., Gravdahl, J. T., & Børhaug, E. (2012). Path following of underactuated autonomous underwater vehicles in the presence of ocean currents. IEEE Conference on Decision and Control, Maui, HI.

Candeloro, M., Lekkas, A. M., & Sørensen, A. J. (2017). A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. *Control Engineering Practice*, *61*, 41-54.

Candeloro, M., Lekkas, A. M., Sørensen, A. J., & Fossen, T. I. (2013). Continuous Curvature Path Planning using Voronoi diagrams and Fermat's spirals. *IFAC Proceedings Volumes*, *46*(33), 132-137.

Carsten, J., Ferguson, D., & Stentz, A. (2006). 3D Field D: Improved path planning and replanning in three dimensions. IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China.

Casalino, G., Turetta, A., & Simetti, E. (2009). A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields. OCEANS 2009 - Europe, Bremen, Germany.

Chen, C. W., Kouh, J. S., & Tsai, J. F. (2013). Modeling and simulation of an AUV simulator with guidance system. *IEEE Journal of Oceanic Engineering*, *38*(2), 211-225.

Cheng, C., Zhu, D., Sun, B., Chu, Z., Nie, J., & Zhang, S. (2015). Path planning for autonomous underwater vehicle based on artificial potential field and velocity synthesis. IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, NS, Canada.

Cheng, Y., & Zhang, W. (2018). Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing*, *272*, 63-73.

Choe, R., Puig-Navarro, J., Cichella, V., Xargay, E., & Hovakimyan, N. (2016). Cooperative trajectory generation using pythagorean hodograph Bézier curves. *Journal of guidance, control, and dynamics*, *39*(8), 1744-1763.

Coath, G., & Halgamuge, S. K. (2003). A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. Congress on Evolutionary Computation, Canberra, Australia.

Cui, R., Li, Y., & Yan, W. (2016). Mutual information-based multi-AUV path planning for scalar field sampling using multidimensional RRT*. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *46*(7), 993-1004.

Cui, R., Yang, C., Li, Y., & Sharma, S. (2017). Adaptive Neural Network Control of AUVs With Control Input Nonlinearities Using Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *47*(6), 1019-1029.

Dahl, A. R. (2013). *Path planning and guidance for marine surface vessels* Norwegian University of Science and Technology]. Trondheim, Norway.

Dai, R., & Cochran, J. E. (2009). Path planning for multiple unmanned aerial vehicles by parameterized cornu-spirals. American Control Conference, St. Louis, MO.

Dariani, R., Schmidt, S., & Kasper, R. (2014). Optimization based obstacle avoidance. *International Journal of Computer and Information Engineering*, *8*(9), 1567-1572.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, *79*(3), 497-516.

Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., & Jurišica, L. (2014). Path Planning with Modified a Star Algorithm for a Mobile Robot. *Procedia Engineering*, *96*, 59-69.

Duguleana, M., & Mogan, G. (2016). Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems with Applications*, *62*, 104-115.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan.

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. Congress on Evolutionary Computation, La Jolla, CA.

Eberhart, R. C., & Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. Congress on Evolutionary Computation, Seoul, South Korea.

Edwards, J. R., Smith, J., Girard, A., Wickman, D., Lermusiaux, P. F. J., Subramani, D. N., Haley, P. J., Mirabito, C., Kulkarni, C. S., & Jana, S. (2017). Data-driven learning and modeling of AUV operational characteristics for optimal path planning. OCEANS 2017 MTS/IEEE, Aberdeen, UK.

Elbanhawi, M., & Simic, M. (2014). Sampling-based robot motion planning: A review. *IEEE Access*, *2*, 56-77.

Elhoseny, M., Tharwat, A., & Hassanien, A. E. (2018). Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm. *Journal of Computational Science*, *25*, 339-350.

Eng, Y., Teo, K. M., Chitre, M., & Ming Ng, K. (2016). Online system identification of an autonomous underwater vehicle via in-field experiments. *IEEE Journal of Oceanic Engineering*, *41*(1), 5-17.

Farouki, R. T. (2008). *Pythagorean-hodograph curves: Algebra and geometry inseparable*. Springer.

Farouki, R. T., Giannelli, C., Mugnaini, D., & Sestini, A. (2018). Path planning with Pythagorean-hodograph curves for unmanned or autonomous vehicles. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, *232*(7), 1361-1372.

Ferguson, D., & Stentz, A. (2006a). Anytime RRTs. IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China.

Ferguson, D., & Stentz, A. (2006b). Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics*, *23*(2), 79-101.

Ferri, G., Munafò, A., & LePage, K. D. (2018). An autonomous underwater vehicle data-driven control strategy for target tracking. *IEEE Journal of Oceanic Engineering*, *43*(2), 323-343.

Foo, J. L., Knutzon, J., Oliver, J., & Winer, E. (2006). Three-dimensional path planning of unmanned aerial vehicles using particle swarm optimization. 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA.

Fossen, T. I. (1999). *Guidance and Control of Ocean Vehicles*. John Wiley & Sons.

Fraichard, T., & Scheuer, A. (2004). From Reeds and Shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, *20*(6), 1025-1035.

Fu, Y., Ding, M., & Zhou, C. (2012). Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *42*, 511-526.

Fu, Y., Ding, M., Zhou, C., Cai, C., & Sun, Y. (2009). Path planning for UAV based on quantum-behaved particle swarm optimization. Proceedings of SPIE - The International Society for Optical Engineering, Yichang, China.

Fu, Y., Ding, M., Zhou, C., & Hu, H. (2013). Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *43*(6), 1451–1465.

Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M., & Ridao, P. (2015). Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. *Journal of Field Robotics*, *32*(7), 952–983.

Garau, B., Alvarez, A., & Oliver, G. (2006). AUV navigation through turbulent ocean environments supported by onboard H-ADCP. Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA), Orlando, FL.

Garau, B., Bonet, M., Alvarez, A., Ruiz, S., & Pascual, A. (2009). Path planning for autonomous underwater vehicles in realistic oceanic current fields: Application to gliders in the western mediterranean sea. *Journal of Maritime Research*, *6*(2), 5-22.

Gong, D.-w., Zhang, J.-h., & Zhang, Y. (2011). Multi-objective particle swarm optimization for robot path planning in environment with danger sources. *Journal of computers*, *6*(8), 1554-1561.

Haddadin, S., Urbanek, H., Parusel, S., Burschka, D., Roßmann, J., Albu-Schäffer, A., & Hirzinger, G. (2010). Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan.

Hao, Y., Zu, W., & Zhao, Y. (2007). Real-Time Obstacle Avoidance Method based on Polar Coordination Particle Swarm Optimization in Dynamic Environment. 2nd IEEE Conference on Industrial Electronics and Applications, Harbin, China.

Hassani, V., & Lande, S. V. (2018). Path Planning for Marine Vehicles using Bézier Curves. *IFAC-PapersOnLine*, *51*(29), 305-310.

Hernández, E., Carreras, M., Antich, J., Ridao, P., & Ortiz, A. (2011). A topologically guided path planner for an AUV using homotopy classes. IEEE International Conference on Robotics and Automation, Shanghai, China.

Hernández, J. D., Vidal, E., Moll, M., Palomeras, N., Carreras, M., & Kavraki, L. E. (2019). Online motion planning for unexplored underwater environments using autonomous underwater vehicles. *Journal of Field Robotics*, *36*(2), 370–396.

Hochberg, Y., & Tamhane, A. C. (1987). *Multiple Comparison Procedures*. John Wiley & Sons.

Hollinger, G. A., Pereira, A. A., Binney, J., Somers, T., & Sukhatme, G. S. (2016). Learning uncertainty in ocean current predictions for safe and reliable navigation of underwater vehicles. *Journal of Field Robotics*, *33*(1), 47-66.

Hudson, J., & Seto, M. L. (2014). Underway path-planning for an unmanned surface vehicle performing cooperative navigation for UUVs at varying depths. IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL.

Huynh, V. T., Dunbabin, M., & Smith, R. N. (2015). Predictive motion planning for AUVs subject to strong time-varying currents and forecasting uncertainties. IEEE international conference on robotics and automation (ICRA), Seattle, WA.

Jolly, K. G., Kumar, R. S., & Vijayakumar, R. (2009). A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, *57*(1), 23-33.

Koay, T.-B., & Chitre, M. (2013). Energy-efficient path planning for fully propelled AUVs in congested coastal waters. OCEANS 2013 MTS/IEEE, Bergen, Norway.

Kruger, D., Stolkin, R., Blum, A., & Briganti, J. (2007). Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy.

Larson, J., Bruch, M., & Ebken, J. (2006). Autonomous navigation and obstacle avoidance for unmanned surface vehicles. Unmanned systems technology VIII, Orlando, FL.

Lekkas, A. M., Dahl, A. R., Breivik, M., & Fossen, T. I. (2013). Continuous-curvature path generation using Fermat's spiral. *Modeling, Identification and Control*, *34*(4), 183-198.

Lekkas, A. M., & Fossen, T. I. (2014). Integral LOS path following for curved paths based on a monotone cubic Hermite spline parametrization. *IEEE Transactions on Control Systems Technology*, *22*(6), 2287-2301.

Li, G., Hildre, H. P., & Zhang, H. (2020). Toward Time-Optimal Trajectory Planning for Autonomous Ship Maneuvering in Close-Range Encounters. *IEEE Journal of Oceanic Engineering*, *45*(4), 1219-1234.

Likhachev, M., Gordon, G., & Thrun, S. (2004). ARA*: Anytime A* with provable bounds on sub-optimality. Advances in Neural Information Processing Systems, Vancouver, Canada.

Lin, C., Wang, H., Yuan, J., Yu, D., & Li, C. (2019). An improved recurrent neural network for unmanned underwater vehicle online obstacle avoidance. *Ocean Engineering*, *189*, 106327.

Locatelli, M. (2003). A note on the Griewank test function. *Journal of global optimization*, *25*(2), 169-174.

Lolla, T., Lermusiaux, P. F., Ueckermann, M. P., & Haley, P. J. (2014). Time-optimal path planning in dynamic flows using level set equations: theory and schemes. *Ocean Dynamics*, *64*(10), 1373-1397.

Lolla, T., Ueckermann, M., Yiğit, K., Haley, P. J., & Lermusiaux, P. F. (2012). Path planning in time dependent flow fields using level set methods. IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, MN.

Lv, M., Yang, C., & Zhang, S. (2019). Real-time Route Re-planning based on Modified Particle Swarm Optimization Algorithm. IEEE 2nd International Conference on Electronic Information and Communication Technology (ICEICT), Harbin, China.

Ma, T., Li, Y., Jiang, Y., Wang, R., Cong, Z., & Gong, Y. (2018). A dynamic path planning method for terrain-aided navigation of autonomous underwater vehicles. *Measurement Science and Technology*, *29*(9), 095105.

MahmoudZadeh, S., Yazdani, A. M., Sammut, K., & Powers, D. M. (2018). Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms. *Applied Soft Computing*, *70*, 929–945.

McKight, P. E., & Najab, J. (2010). Kruskal-Wallis test. *The corsini encyclopedia of psychology*.

McMahon, J., & Plaku, E. (2016). Mission and Motion Planning for Autonomous Underwater Vehicles Operating in Spatially and Temporally Complex Environments. *IEEE Journal of Oceanic Engineering*, *41*(4), 893-912.

Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation methods. *Evolutionary programming*, *4*, 135-155.

Mirjalili, S., Song Dong, J., & Lewis, A. (2020). Ant colony optimizer: Theory, literature review, and application in AUV path planning. In *Nature-Inspired Optimizers* (Vol. 811, pp. 7-21). Springer, Cham.

Modares, H., & Sistani, M.-B. N. (2011). Solving nonlinear optimal control problems using a hybrid IPSO–SQP algorithm. *Engineering Applications of Artificial Intelligence*, *24*(3), 476-484.

Mohamed, A. Z., Lee, S. H., Hsu, H. Y., & Nath, N. (2012). A faster path planner using accelerated particle swarm optimization. *Artificial Life and Robotics*, *17*(2), 233-240.

Mühlenbein, H., Schomisch, M., & Born, J. (1991). The parallel genetic algorithm as function optimizer. *Parallel computing*, *17*(6-7), 619-632.

Naeem, W., Irwin, G. W., & Yang, A. (2012). COLREGs-based collision avoidance strategies for unmanned surface vehicles. *Mechatronics*, *22*(6), 669-678.

Newman, P. M. (2008). *MOOS - Mission Orientated Operating Suite (OUEL Report)*. Department of Engineering Science, University of Oxford.

Panda, M., Das, B., Subudhi, B., & Pati, B. B. (2020). A comprehensive review of path planning algorithms for autonomous underwater vehicles. *International Journal of Automation and Computing*, 1-32.

Park, C., Pan, J., & Manocha, D. (2013). Real-time optimization-based planning in dynamic environments using GPUs. IEEE International Conference on Robotics and Automation, Karlsruhe, Germany.

Parkinson, C., Bertozzi, A. L., & Osher, S. J. (2020). A Hamilton-Jacobi Formulation for Time-Optimal Paths of Rectangular Nonholonomic Vehicles. 2020 59th IEEE Conference on Decision and Control (CDC), Jeju, South Korea.

Paull, L., Saeedi, S., Seto, M., & Li, H. (2013). Sensor-driven online coverage planning for autonomous underwater vehicles. *IEEE/ASME Transactions on Mechatronics*, *18*(6), 1827-1838.

Pereira, A. A., Binney, J., Hollinger, G. A., & Sukhatme, G. S. (2013). Risk‐aware path planning for autonomous underwater vehicles using predictive ocean models. *Journal of Field Robotics*, *30*(5), 741-762.

Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., & Lane, D. (2007). Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, *23*(2), 331-341.

Petres, C., Romero-Ramirez, M.-A., & Plumet, F. (2011). Reactive path planning for autonomous sailboat. 15th International Conference on Advanced Robotics (ICAR), Tallinn, Estonia.

Piegl, L., & Tiller, W. (2012). *The NURBS book*. Springer Science & Business Media.

Prestero, T. J. (2001). *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle* [Master's thesis, Massachusetts Institute of Technology].

Pyper, W. (2018). Yellow submarine prepares for first Antarctic mission. *Australian Antarctic Magazine* (35), 12–13. https://search.informit.org/doi/10.3316/informit.216707160549452

Qian, Q., Tokgo, M., Kim, C., Han, C., Ri, J., & Song, K. (2015). A hybrid improved quantum-behaved particle swarm optimization algorithm using adaptive coefficients and natural selection method. Advanced Computational Intelligence (ICACI), 2015 Seventh International Conference, Wuyi, China.

Qin, Y., Sun, D., Li, N., & Cen, Y. (2004). Path planning for mobile robot using the particle swarm optimization with mutation operator. International Conference on Machine Learning and Cybernetics, Shanghai, China.

Rao, D., & Williams, S. B. (2009). Large-scale path planning for underwater gliders in ocean currents. Australasian Conference on Robotics and Automation (ACRA), Sydney, Australia.

Raphael, B., & Smith, I. F. (2003). *Fundamentals of computer-aided engineering*. John Wiley & Sons.

Redding, J., Amin, J., Boskovic, J., Kang, Y., Hedrick, K., Howlett, J., & Poll, S. (2007). A Real-Time Obstacle Detection and Reactive Path Planning System for Autonomous Small-Scale Helicopters. AIAA Guidance, Navigation and Control Conference and Exhibit, Hilton Head, SC.

Reeds, J., & Shepp, L. (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, *145*(2), 367-393.

Saska, M., Macas, M., Preucil, L., & Lhotska, L. (2006). Robot path planning using particle swarm optimization of Ferguson splines. IEEE Conference on Emerging Technologies and Factory Automation, Prague, Czech Republic.

Shanmugam, G. (2020). *Mass transport, gravity flows, and bottom currents: Downslope and alongslope processes and deposits*. Elsevier.

Shanmugavel, M., Tsourdos, A., White, B., & Żbikowski, R. (2010). Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs. *Control Engineering Practice*, *18*(9), 1084-1092.

Shanmugavel, M., Tsourdos, A., Zbikowski, R., & White, B. (2007). 3D path planning for multiple UAVs using Pythagorean hodograph curves. AIAA Guidance, Navigation and Control Conference and Exhibit, Hilton Head, SC.

Shi, Y., & Eberhart, R. C. (1998). A modified particle swarm optimizer. IEEE International Conference on Evolutionary Computation, Anchorage, AK.

Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. Congress on Evolutionary Computation, Washington, DC.

Singh, Y., Sharma, S., Sutton, R., Hatton, D., & Khan, A. (2018). A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a

maritime environment containing dynamic obstacles and ocean currents. *Ocean Engineering*, *169*, 187-201.

Song, B., Wang, Z., Zou, L., Xu, L., & Alsaadi, F. E. (2017). A new approach to smooth global path planning of mobile robots with kinematic constraints. *International Journal of Machine Learning and Cybernetics*, 1-13.

Song, L., Mao, Y., Xiang, Z., Zhou, Y., & Du, K. (2015). A study on path planning algorithms based upon Particle Swarm Optimization. *Journal of Information and Computational Science*, *12*(2), 673-680.

Spain, E., Gwyther, D., & King, P. (2019). Submarine ventures under Sørsdal Glacier. *Australian Antarctic Magazine*, (36), 18–19. https://search.informit.org/doi/10.3316/informit.520443225029371

Sun, B., & Zhu, D. (2016). Three dimensional D∗ lite path planning for autonomous underwater vehicle under partly unknown environment. 12th World Congress on Intelligent Control and Automation (WCICA), Guilin, China.

Sun, B., Zhu, D., & Yang, S. X. (2018). An optimized fuzzy control algorithm for three-dimensional AUV path planning. *International Journal of Fuzzy Systems*, *20*(2), 597–610.

Sun, J., Feng, B., & Xu, W. (2004). Particle swarm optimization with particles having quantum behavior. Congress on Evolutionary Computation, Portland, OR.

Sun, J., Lai, C. H., & Wu, X. J. (2012). *Particle swarm optimisation classical and quantum perspectives*. CRC Press.

Takei, R., & Tsai, R. (2013). Optimal Trajectories of Curvature Constrained Motion in the Hamilton–Jacobi Formulation. *Journal of Scientific Computing*, *54*(2-3), 622-644.

Taleshian, T., & Minagar, S. (2015). Motion planning for an autonomous underwater vehicle. 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran.

Tanakitkorn, K., Wilson, P. A., Turnock, S. R., & Phillips, A. B. (2014). Grid-based GA path planning with improved cost function for an over-actuated hover-capable AUV. IEEE/OES Autonomous Underwater Vehicles (AUV), Oxford, MS.

Techy, L., & Woolsey, C. A. (2009). Minimum-time path planning for unmanned aerial vehicles in steady uniform winds. *Journal of guidance, control, and dynamics*, *32*(6), 1736-1746.

Tsourdos, A., White, B., & Shanmugavel, M. (2010). *Cooperative path planning of unmanned aerial vehicles* (Vol. 32). John Wiley & Sons.

Vasile, C. I., & Belta, C. (2014). Reactive sampling-based temporal logic path planning. IEEE International Conference on Robotics and Automation (ICRA), Hong Kong.

Wang, H., Zhou, H., & Yao, H. (2016). Research on autonomous planning method based on improved quantum Particle Swarm Optimization for Autonomous Underwater Vehicle. OCEANS 2016 MTS/IEEE, Monterey, CA.

Wang, N., Jin, X., & Er, M. J. (2019). A multilayer path planner for a USV under complex marine environments. *Ocean Engineering*, *184*, 1-10.

Witt, J., & Dunbabin, M. (2008). Go with the flow: Optimal AUV path planning in coastal environments. Australasian Conference on Robotics and Automation, Canberra, Australia.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

Xue, T., Li, R., Tokgo, M., Ri, J., & Han, G. (2017). Trajectory planning for autonomous mobile robot using a hybrid improved QPSO algorithm. *Soft Computing*, *21*(9), 2421-2437.

Yang, X. S., Deb, S., & Fong, S. (2011). Accelerated particle swarm optimization and support vector machine for business optimization and applications. International Conference on Networked Digital Technologies, Macau, China.

Yao, X., Wang, F., Wang, J., & Wang, X. (2018). Bilevel optimization-based time-optimal path planning for AUVs. *Sensors*, *18*(12), 4167.

Yao, X., Wang, X., Wang, F., & Zhang, L. (2020). Path following based on waypoints and real-time obstacle avoidance control of an autonomous underwater vehicle. *Sensors*, *20*(3), 795.

Youakim, D., & Ridao, P. (2018). Motion planning survey for autonomous mobile manipulators underwater manipulator case study. *Robotics and Autonomous Systems*, *107*, 20–44.

Zeng, Z., Sammut, K., He, F., & Lammas, A. (2012). Efficient path evaluation for AUVs using adaptive B-spline approximation. OCEANS 2012 MTS/IEEE, Hampton Roads, VA.

Zeng, Z., Sammut, K., Lammas, A., He, F., & Tang, Y. (2015). Efficient path re-planning for AUVs operating in spatiotemporal currents. *Journal of Intelligent & Robotic Systems*, *79*(1), 135–153.

Zeng, Z., Sammut, K., Lian, L., He, F., Lammas, A., & Tang, Y. (2016). A comparison of optimization techniques for AUV path planning in environments with ocean currents. *Robotics and Autonomous Systems*, *82*, 61–72.

Zhan, Z. H., Zhang, J., Li, Y., & Chung, H. S. H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(6), 1362-1381.

Zhang, C. B., Gong, Y. J., Li, J. J., & Lin, Y. (2014). Automatic path planning for autonomous underwater vehicles based on an adaptive differential evolution. 2014 Annual Conference on Genetic and Evolutionary Computation, Vancouver, Canada.

Zhang, H., Yang, L., Gao, Z., & Cao, Y. (2011). The dynamic path planning research for mobile robot based on artificial potential field. International Conference on Consumer Electronics, Communications and Networks (CECNet), Xianning, China.

Zhang, H. H., Gong, L., Tao, C., Lu, W., & Xun, Z. (2016). Global path planning methods of UUV in coastal environment. IEEE International Conference on Mechatronics and Automation (ICMA), Harbin, China.

Zhang, W. J., & Xie, X. F. (2003). DEPSO: hybrid particle swarm with differential evolution operator. IEEE International Conference on Systems, Man and Cybernetics, Washington, DC.

Zhong, W. M., Li, S. J., & Qian, F. (2008). θ-PSO: a new strategy of particle swarm optimization. *Journal of Zhejiang University-SCIENCE A*, *9*(6), 786-790.

Zhou, H., Zeng, Z., & Lian, L. (2018). Adaptive re-planning of AUVs for environmental sampling missions: a fuzzy decision support system based on multi-objective particle swarm optimization. *International Journal of Fuzzy Systems*, *20*(2), 650-671.

Zu, W., Li, G., & Qi, Z. (2008). Study on path planning method based on improved particle swarm optimization. *Journal of Projectiles, Rockets, Missiles and Guidance*, *28*(4), 68-70.

This page is intentionally left blank.

Appendix A.

Specifications of the REMUS 100 AUV

Physical specifications	
Length	1.7 m
Diameter	190 mm
Weight in air	36 kg
Maximum depth	100 m
Endurance	≤ 12 hours (at 1.5 m/s)
Velocity range	0 – 2.6 m/s
Energy storage	1.5 kWh, internal rechargeable lithium-ion batteries
Propulsion	Direct drive DC brushless motor to open 3-bladed propeller
Navigation	Long Baseline (LBL), Doppler-assisted dead reckoning, Inertial Navigation System (INS), GPS
Actuators	Cruciform rudders and a three-bladed propeller

This page is intentionally left blank.

Appendix B.

Specifications of the "*nupiri muka*" AUV

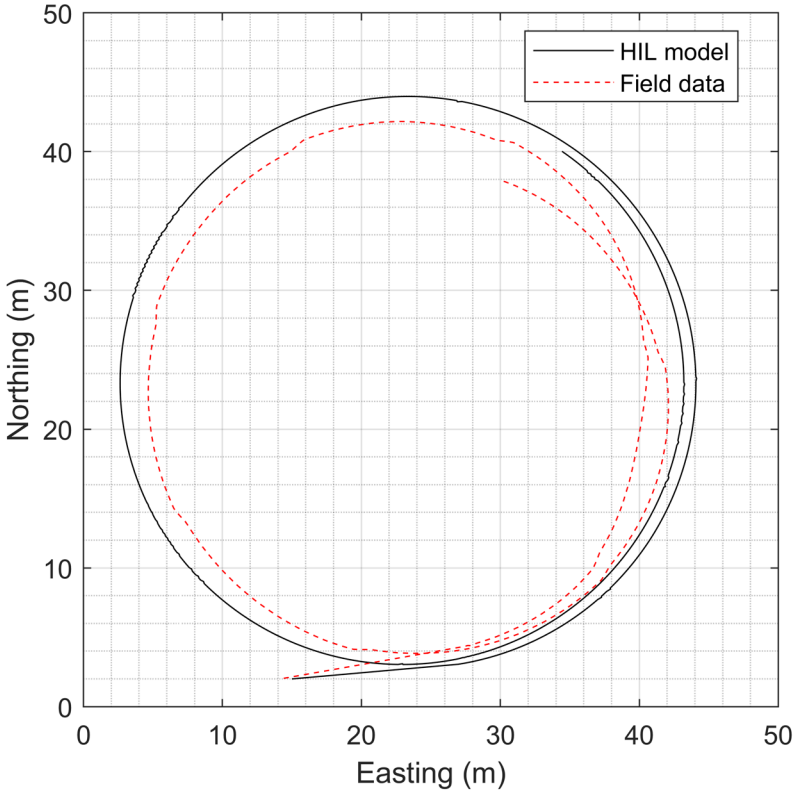
Physical specifications	
Length	7.5 m
Diameter	740 mm
Weight in air	2000 kg
Maximum depth	5000 m
Endurance	≤ 40 hours
Velocity range	0 – 2.2 m/s
Actuators	A pair of hydroplanes, X-form rudders and a two-bladed propeller
Specifications of onboard computer	
CPU	Intel Celeron Processor J1900 2.0 GHz with 2 MB L2 cache
System chipset	Intel Celeron J1900 SoC
BIOS	AMI 16 Mbit Flash BIOS
System memory	DDR3L 1333 MHz up to 8 GB One 204-pin SO DIMM socket
Watchdog timer	Single-chip Watchdog 255-level interval timer, setup by software
I/O interface	2 x RS232, 2 x RS232/422/485
USB	3 x USB 2.0, 1 x USB 3.0 compliant ports
Audio	High-definition Audio, Line-in, Line out, Mic-in
Storage	1 x mSATA and 1 x high capacity 2.5" SATA HDD (up to 12.5 mm height)
Expansion interface	Supports1 x MiniPCIe with SIM holder Supports 1 x iDoor expansion
Software API	Advantech iManager/SUSI 4.0 and SUSIAccess – Remote Device Management technology

This page is intentionally left blank.

Appendix C.

Validation of the "*nupiri muka*" HIL model

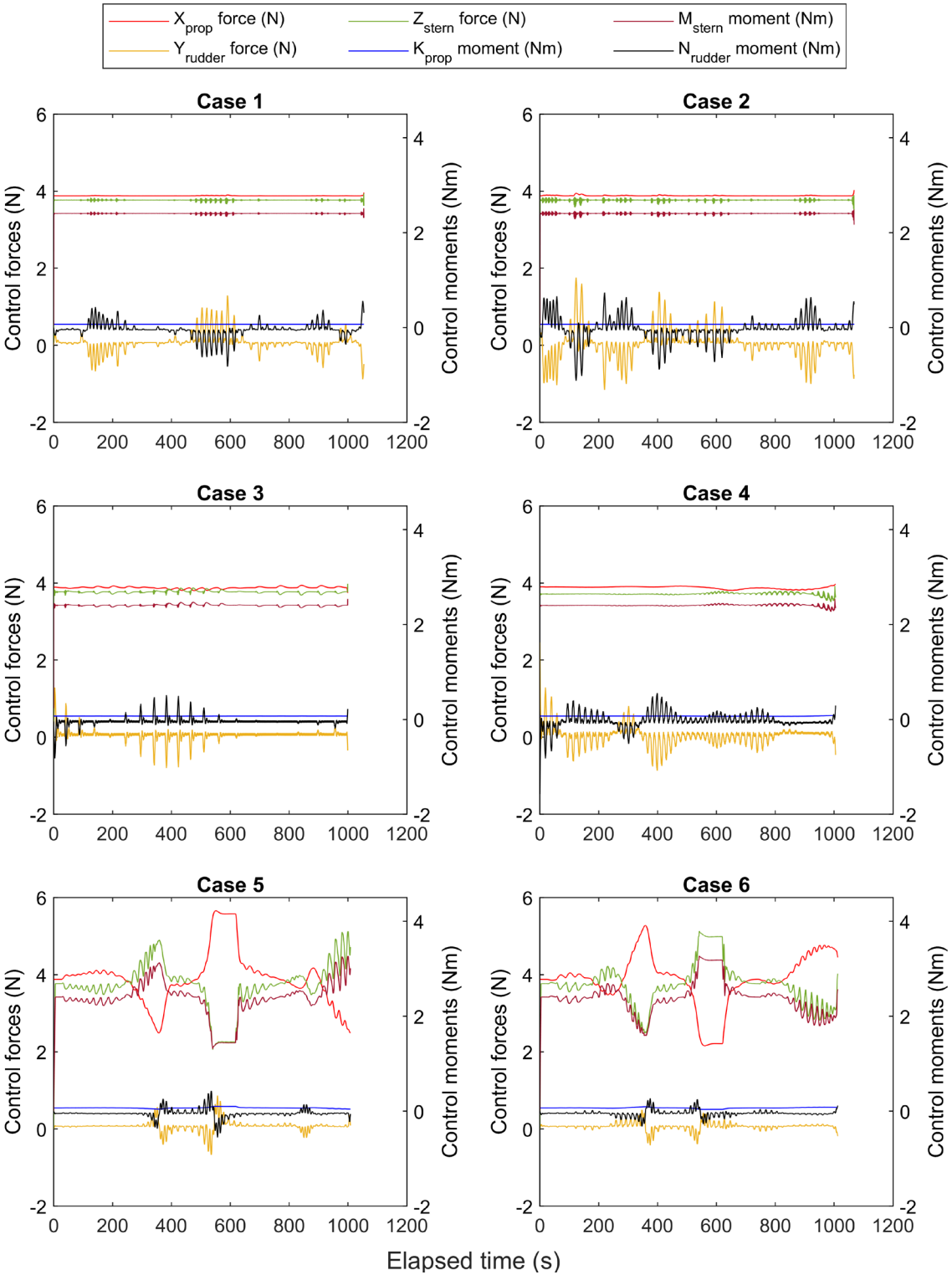
The vehicle model used in the HIL setup was validated against the field data of the *nupiri muka* AUV through turning circle tests, which required the vehicle to enter a steady turn at a constant speed. The experimental data were collected in the field with the AUV conducting a full-rudder (23°) circle at a vehicle speed of 1.5 m/s. The turning circle of the vehicle model was obtained from the HIL setup using the same rudder angle and vehicle speed. The two trajectories produced similar turning radius with a discrepancy of less than 5%.



This page is intentionally left blank.

Appendix D.

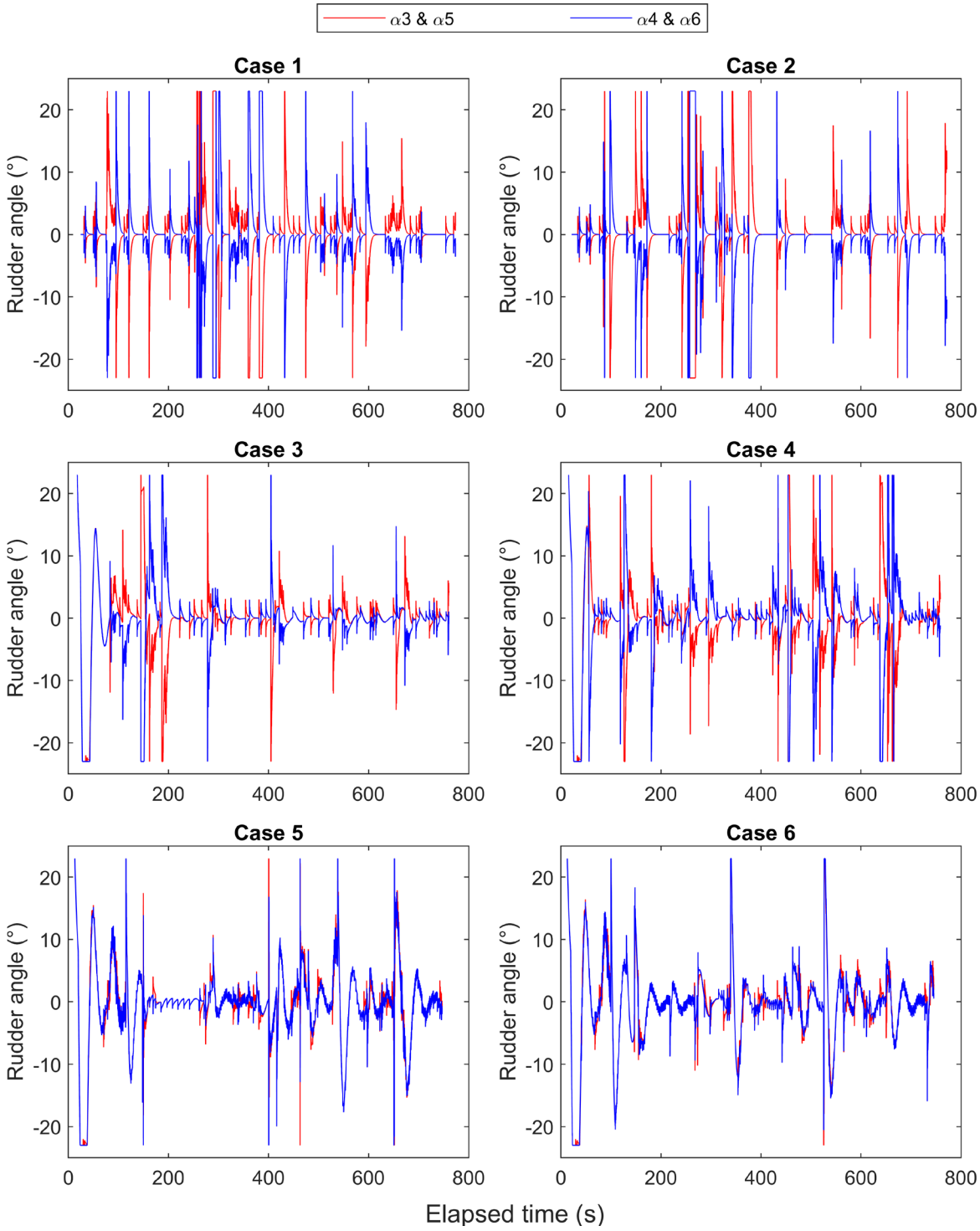
Control forces and moments of the REMUS 100 AUV model



This page is intentionally left blank.

Appendix E.

Control signals of the "*nupiri muka*" AUV



This page is intentionally left blank.

Appendix F.

Publications

(Continued.)

Performance evaluation of particle swarm intelligence-based optimization techniques in a novel AUV path planner

Hui Sheng Lim
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
hui.lim@utas.edu.au

Shuangshuang Fan
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
shuangshuang.fan@utas.edu.au

Christopher K.H. Chin
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
c.chin@utas.edu.au

Shuhong Chai
National Centre for Maritime Engineering and Hydrodynamics
Australian Maritime College
University of Tasmania
Launceston, Australia
shuhong.chai@utas.edu.au

***Abstract*—Over years of development, many optimization techniques have been proposed for the path planning of an autonomous underwater vehicle (AUV). The development in swarm intelligence optimization, particularly the particle swarm optimization (PSO) algorithm, has significantly improved the performance of the AUV path planner. This study presents 11 variants of particle swarm intelligence (PSI)-based algorithms, which were applied to evaluate their performances in solving the optimal path planning problem of an AUV operating in 2D and 3D ocean environments with obstacles and non-uniform currents. Throughout the structure of the optimization problem, the practicability of the path planning algorithms was considered by taking into account the physical limitations of the AUV actuation. To compare the performances of these PSI-based algorithms, extensive Monte Carlo simulations were conducted to evaluate these algorithms based on their respective solution qualities, stabilities and computational efficiencies. Ultimately, the strengths and weaknesses of these algorithms were comprehensively analyzed, to identify the most appropriate optimization algorithm for AUV path planning in dynamic environments.**

Keywords—Autonomous Underwater Vehicle; Path planning; Optimization; Swarm intelligence; Particle Swarm Optimization

I. INTRODUCTION

AUVs are unmanned underwater vehicles that can be remotely programmed to conduct various missions. To date, many efforts have been made to enable the operation of AUVs in more dynamic and constrained environments. The exploration of AUVs in highly dynamic regions has several technical issues, particularly for its path planning. An optimum AUV path planner should be able to determine a path that safely guides the AUV from a starting point to a target under rapid-changing dynamic environments, based on either minimum time or energy cost criterion [1].

Planning the path for the AUVs is essentially a multimodal optimization problem. Developing the algorithms for AUV path planning faces several intrinsic difficulties, particularly in balancing the computational requirements and performance of the path planner. For a high-dimensional problem space, the computational requirement of the algorithms could escalate exponentially. A general path planning approach is to simplify the 3D environment into a 2D space, in order to reduce the computational time and the memory requirement [1]. However, this compromises the performance of the path planner due to the reduced amount of 3D information available, such as current profiles, bathymetry, and obstacles in the ocean environment. A recent comparison study in [2] for the existing path optimization techniques proved the superiority of evolutionary algorithms, particularly the evolutionary particle swarm intelligence (PSI)-based algorithms, which were found to be remarkably robust and efficient for solving high-dimensional path planning problem.

Particle swarm optimization (PSO) and its most significant variant, the quantum-behaved particle swarm optimization (QPSO) are extensively used in various optimization problems ever since their emergence in 1995 and 2004 respectively due to their fine search abilities and easy implementations [3]. In recent years, many strategies that modified the PSO and QPSO algorithms have been proposed to improve their performances in path planning of various autonomous systems. Each of these variants of the algorithms was claimed to have different extents of improvement over the original PSO and QPSO. Nevertheless, there is a lack of systematic methods to evaluate the performance of these algorithms. It is crucial to present a study that compares and reviews these algorithms for the required application.

Therefore, this paper aims to provide a comprehensive evaluation study on these algorithms through the application in AUV path planning. A novel path planner based on an AUV was developed and integrated with different PSI-based algorithms. To evaluate the performance of these algorithms, path planning scenarios with multiple obstacles and non-uniform currents was simulated in 2D and 3D domains. Although the actual AUV operates in a 3D ocean field, 3D path planning simulations are not widely discussed in the majority of other literature due to the complexity of high dimensional path planning problem. It is critical to apply the path planning algorithms in 3D space to assess the effectiveness of the algorithms under realistic conditions. Extensive Monte Carlo simulations were conducted to analyse the performances of different PSI-based algorithms.

The rest of this paper is arranged as follows. Section II provides a literature review on the basic PSO, QPSO and their variants. The path planning problem is formulated in Section III. Section IV presents the simulation setup, results and discussion. Lastly, Section V concludes the paper with future directions.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©2018 IEEE

II. PSO, QPSO AND THEIR VARIANTS

This section presents an overview of various PSI-based algorithms, including the basic PSO, QPSO, and their variants. The variants of PSO and QPSO were classified based on the methods of modification used to improve their performances.

A. PSO Algorithm

PSO is a heuristic optimization algorithm introduced by [4] based on the inspiration from the analogues of cognitive abilities and social interaction in animals. [5, 6] are some examples of pioneering works of PSO applications in path planning. PSO consists of particles that move within a multidimensional search space to search for potential solutions, which can be represented by the particles' positions. The particles' velocities are updated by the particle's own experience (cognitive behaviour) and the swarm's experience (social behaviour) to vary their positions.

In a standard PSO algorithm consisting of N particles with D number of dimensions for solving a cost evaluation function f, the position vector of the i^{th} particle at t^{th} iteration is denoted as:

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ \ldots \ , x_{i,D}^t \right], \quad i \in \{1, 2, ..., N\} \tag{1}$$

Based on its previous best position, *pbest* and global best position in the swarm, *gbest*, the velocity V and the position X of the i^{th} particle at $(t+1)^{\text{th}}$ iteration are updated as follows:

$$V_i^{t+1} = w \cdot V_i^t + C_1 \cdot r_1^t \cdot (pbest_i^{\ t} - X_i^t) + C_2 \cdot r_2^t \cdot (gbest^{\ t} - X_i^t) \tag{2}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{3}$$

$$pbest_i^{\ t} = \begin{cases} pbest_i^{t-1}, & if\ f(X_i^t) \geq f(pbest_i^{t-1}) \\ X_i^t, & if\ f(X_i^t) < f(pbest_i^{t-1}) \end{cases} \tag{4}$$

$$gbest^{\ t} = \arg\min \left[f(pbest_i^{\ t}) \right] \tag{5}$$

In (2), r_1 and r_2 are random positive numbers that are less than 1.0. C_1 and C_2 are the acceleration coefficients for cognitive and social components respectively; they are both set to 2.0 for most applications [3]. w is the inertia weight for balancing the particle global exploration and local exploitation to improve the performance. The common strategy is to set w at an initial value of 0.9, and linearly decrease to 0.4 during the iteration [3].

B. QPSO Algorithm

Inspired by quantum mechanics and PSO, [7] proposed the QPSO algorithm, which assumes the particles to have quantum behaviour. QPSO algorithm is well known to be an improved version of PSO. Its application in path planning was pioneered by [8, 9]. In QPSO, the position of the i^{th} particle is given as:

$$X_i^{t+1} = \begin{cases} \varphi_i^t \cdot pbest_i^{\ t} + (1 - \varphi_i^t) \cdot gbest^{\ t} \\ \quad + \beta \cdot \left| mbest^{\ t} - X_i^{\ t} \right| \cdot \ln\left(1/u_i^t\right), & if\ u \geq 0.5 \\ \varphi_i^t \cdot pbest_i^{\ t} + (1 - \varphi_i^t) \cdot gbest^{\ t} \\ \quad - \beta \cdot \left| mbest^{\ t} - X_i^{\ t} \right| \cdot \ln\left(1/u_i^t\right), & if\ u < 0.5 \end{cases} \tag{6}$$

$$mbest^{\ t} = \sum_{i=1}^{N} pbest_i^t / N \tag{7}$$

where u and φ are random positive numbers that are less than 1. β is the contraction-expansion (CE) coefficient, and *mbest* is the mean best position which is defined as the average of personal best positions of all particles as shown in (7). When applying the QPSO algorithm, β is the most critical parameter for controlling the algorithm performance. A linearly decreasing β from $\beta_{\max}$ of 1.0 to $\beta_{\min}$ of 0.5 is suggested for most applications [3].

C. Variants of PSO and QPSO

1) Improvement by Controlling Parameters

In PSI-based algorithms, the equation coefficients are the most critical parameters for controlling the performance. In PSO, w, C_1 and C_2 must be controlled to balance the particles' global exploration and local exploitation. Reference [10] proposed an adaptive PSO, which uses an evolutionary factor f as an indicator representing the particles' evolutionary state to control the equation coefficients. The adaptive PSO was applied in solving a robotic path planning problem by [11]. To determine the evolutionary factor f, the mean distance d_i of the i^{th} particle to other particles is calculated using (8). f is then given by (9).

$$d_i = \frac{1}{N-1} \sum_{j=1,\, j \neq i}^{N} \sqrt{\sum_{k=1}^{D} \left(x_i^k - x_j^k \right)^2} \tag{8}$$

$$f = \left(d_g - d_{\min} \right) \Big/ \left(d_{\max} - d_{\min} \right) \quad \in [0, 1] \tag{9}$$

where d_g is the mean distance of the global best particle, $d_{\min}$ and $d_{\max}$ are the minimum and maximum of mean distances respectively. f varies from 1 - 0 as the particles move from global exploration to local exploitation phase. w can be calculated from f using (10), while C_1 and C_2 can be adapted using (11).

$$w = 1 \Big/ \left(1 + 1.5 e^{-2.6 f} \right) \quad \in [0.4, 0.9] \tag{10}$$

$$\begin{aligned} C_1 &= 0.8 + 2e^{-|f - 0.5|} \\ C_2 &= 3.2 - 2e^{-|f - 0.5|}, \quad \text{where } C_1 + C_2 = 4 \end{aligned} \tag{11}$$

The only coefficient that needs to be controlled in QPSO is β. To adapt β with the particle evolution, [12] proposed Dynamic-Weighted QPSO (DWQPSO) to solve an AUV path planning problem. In DWQPSO, β is controlled by classifying the particles based on their fitness F_i. The global best fitness is denoted as F_{gbest}, and the average of all particles fitness values is F_{avg}. The mean of fitness values that are above average (better than F_{avg}) is denoted as F_{good}. The categories of the particles are:

- $F_i \geq F_{avg}$: For these particles, the global exploration of the particles should be boosted with a higher β using (12).

$$\beta_1 = 1.5 - 1 \Big/ \left(1 + 1.5 \cdot e^{(F_{gbest} - F_{good})} \right) \tag{12}$$

- $F_{good} < F_i < F_{avg}$: Linear decreasing model in (13) is used to balance between global exploration and local exploitation.

$$\beta_2 = \beta_{\max} - (t / MaxIt) \cdot (\beta_{\max} - \beta_{\min}) \tag{13}$$

- $F_i \leq F_{good}$: These particles should focus on local exploitation to find the optimal solution. Thus, β is updated as follows.

$$\beta_3 = (\beta_2 - 0.5) \cdot \left| \left(F_i - F_{gbest} \right) \Big/ \left(F_{good} - F_{gbest} \right) \right| \tag{14}$$

2) Improvement by Novel Update Equation

Some studies introduced strategies to improve the algorithm by modifying the position and velocity update equations. In [13],

an accelerated PSO was proposed by simplifying the update equation in PSO. It was successfully applied in developing a fast and simple path planner by [14]. The accelerated PSO disregards the particles' personal best positions and focuses on the global best position using a simple update equation as shown in (15).

$$X_i^{t+1} = 0.5X_i^t + 0.5gbest^t + e^{-0.97t} \cdot \left(r_i^t - 0.5\right) \quad (15)$$

where r is a random number with a value ranging from 0 to 1.0.

In [15], a phase angle-encoded PSO (θ-PSO) was proposed by mapping the position vectors into phase angle vectors through (16), while the increment of phase angle replaces the velocity vectors. The phase angle vector θ of the i^{th} particle at $(t+1)^{\text{th}}$ iteration and its increment $\Delta\theta$ are given by (17) and (18).

$$X_i^t = \left[\left(X_{\max} - X_{\min}\right) \cdot \sin\left(\theta_i^t\right) + X_{\max} + X_{\min}\right] / 2 \quad (16)$$

$$\Delta\theta_i^{t+1} = w \cdot \Delta\theta_i^t + C_1 \cdot r_1^t \left(pbest_i^t - \theta_i^t\right) + C_2 \cdot r_2^t \left(gbest^t - \theta_i^t\right) \quad (17)$$

$$\theta_i^{t+1} = \theta_i^t + \Delta\theta_i^{t+1} \quad \in [-\pi/2, \pi/2] \quad (18)$$

Inspired by [15], θ-QPSO was proposed by [16], who applied a similar phase angle mapping in QPSO to develop an Unmanned Aerial Vehicle (UAV) path planner, and proved that θ-QPSO has better performance than θ-PSO. θ-QPSO only computes for the vector θ as shown in (19).

$$\begin{aligned}\theta_i^{t+1} = {} & \varphi_i^t \cdot pbest_i^t + \left(1 - \varphi_i^t\right) \cdot gbest^t \\ & \pm \beta \cdot \left|mbest^t - \theta_i^t\right| \cdot \ln\left(1/u_i^t\right) \quad \in [-\pi/2, \pi/2]\end{aligned} \quad (19)$$

3) Improvement by Hybrid Method

Hybridization was used to combine the beneficial feature of other optimization techniques with PSO or QPSO algorithm. In [17], PSO was combined with differential evolution (DE) to form DEPSO. DEPSO increases swarm diversity without altering the original particle swarm dynamics. Based on the inspiration from DEPSO, [18] applied the hybridization concept in QPSO to propose DEQPSO, and successfully used both DEPSO and DEPQSO for UAV path planning. In DEPSO and DEQPSO, the conventional position update operation is carried out, followed by a successive DE operation as described below.

- Mutation: A mutated vector U is generated using (20).

$$U_i^t = gbest^t + \left[\left(pbest_{i1}^t - pbest_{i2}^t\right) + \left(pbest_{i3}^t - pbest_{i4}^t\right)\right] / 2 \quad (20)$$

  where i_1, i_2, i_3 and i_4 are randomly selected particle indices and $i_1 \neq i_2 \neq i_3 \neq i_4 \neq gbest$.

- Crossover: A trial vector T is generated to increase the diversity, by conducting crossover between the mutated vector and the personal best position as shown in (21).

$$\begin{aligned} T_i^t &= \left[t_{i,1}^t, \ldots, t_{i,j}^t, \ldots, t_{i,D}^t\right] \\ t_{i,j}^t &= \begin{cases} u_{i,j}^t, & if\ r_j \leq 0.85 \text{ or } j = r \\ pbest_{i,j}^t, & if\ r_j > 0.85 \text{ or } j \neq r \end{cases} \end{aligned} \quad (21)$$

  where r_j is a random number ranging from 0 to 1.0, and r is a random integer ranging from 1 to D.

- Selection: A greedy selection is used to decide whether the trial vector T should replace the current position X in $(t+1)$th iteration. X will only be replaced if T has better fitness value.

4) Improvement by Combination of Multiple Approaches

Some studies improved the algorithm by applying more than one method. Reference [19] proposed IPSO-SQP algorithm, in which an improved PSO (IPSO) with adaptive inertia weight was combined with Sequential Quadratic Programming (SQP) algorithm. SQP has a strong searching ability for the local optimum solution, although its solution quality is highly dependent on the initial solution. SQP was integrated into PSO to accelerate the local exploitation phase. In IPSO-SQP, the inertia weight w is controlled adaptively according to (22).

$$w_i^t = 1/\left[1 + e^{-F(pbest_i^t)/gbest1}\right] \quad (22)$$

When the change in global best fitness between iterations in IPSO-SQP is less than a predefined value, SQP can be initialized using the global best solution from the IPSO operation. The final solution is updated using a greedy selection method, which allows the SQP solution to replace the solution from IPSO only if the SQP solution is better. IPSO-SQP was applied in solving the motion planning of a REMUS AUV by [20].

A hybrid QPSO algorithm, LTQPSO was proposed by [21]. LTQPSO use individual particle evolutionary rates and swarm dispersion as the control parameters for its novel local attractor and position update equations as shown in (23) and (24).

$$p_i^t = gbest^t + ip_i^t \cdot r \cdot \left(pbest_i^t - gbest^t\right) \quad (23)$$

$$X_i^{t+1} = \begin{cases} p_i^t + \beta \cdot \left|mbest^t - gs^t \cdot X_i^t\right| \cdot \ln\left(1/u_i^t\right), & if\ u \geq 0.5 \\ p_i^t - \beta \cdot \left|mbest^t - gs^t \cdot X_i^t\right| \cdot \ln\left(1/u_i^t\right), & if\ u < 0.5 \end{cases} \quad (24)$$

where ip_i^t and gs^t are the control parameters for evolutionary rate and swarm dispersion, as given in (25) and (26) respectively.

$$ip_i^t = F(gbest^t)/F(pbest_i^t) \quad \in [0,1] \quad (25)$$

$$gs^t = \left[\frac{\sigma(pbest_{i,1}^t)}{\sigma(X_{i,1}^t)}, \frac{\sigma(pbest_{i,2}^t)}{\sigma(X_{i,2}^t)}, \ldots, \frac{\sigma(pbest_{i,D}^t)}{\sigma(X_{i,D}^t)}\right] \quad (26)$$

For each iteration in LTQPSO, natural selection is conducted after the standard QPSO operation. Natural selection sorts the particles according to their personal best fitness and replaces those of the worst fitness with those of the best. The natural selection operator increases the evolutionary rate of the entire swarm by eliminating the least desirable solutions, leading to faster global convergence. LTQPSO was proven by [1] to have good performance for robotic path planning in 2D environments.

III. Problem Formulation for AUV Path Planning

A. Path Formulation

In an AUV path planner, the optimal path among a group of potential paths for the AUV to travel toward a target location is required to be determined. Each potential path can comprise a series of nodes from the start point to the endpoint. Optimizing the coordinates of path nodes can produce the optimal path. The start and end points should not be involved in optimization as all potential paths can share the same start and end locations.

Each potential path solution for the problem was modelled as an individual particle in the swarm. The swarm population can be denoted by a matrix $X = [X_1, X_2, \ldots, X_N]^{\text{T}}$, where X is the

particle's position vector and N is the total number of particles. The entries of the position vector represent the coordinates of the path nodes. Assuming a path consists of n+2 nodes including the start and end points, the number of nodes involved in the optimization is n. To record the coordinates of n nodes, the position vector of a particle in 2D has 2n dimensions, while a particle in 3D has 3n dimensions. The position vector of the i^{th} particle at t^{th} iteration for 3D can be given as follows:

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, ..., x_{i,n}^t, x_{i,n+1}^t, ..., x_{i,3n}^t \right], \quad i \in \{1, 2, ..., N\} \quad (27)$$

Based on the path nodes including the start and end points, B-spline geometry was used to construct the AUV path. The path nodes can act as the control points for the B-spline curve according to the curve function in (28), which gives output vector $P(u)$ representing a B-spline curve with k+1 order in the form of discretised waypoints. Given that the total number of control points is n+2, the number of piecewise polynomials is one less than the number of control points, which is n+1.

$$P(u) = \sum_{i=0}^{n+1} x_i \, B_{i,k}(u), \qquad i \in \{0, 1, 2, ..., n+1\} \quad (28)$$

where x_i denotes the control points, u is the non-decreasing knot sequence contained in a knot vector $U = [u_0, \ldots, u_i, \ldots, u_{n+k+2}]$, and $B_{i,k}(u)$ represents the piecewise polynomial basis functions of k degree defined by Cox de Boor recursion [22] as follows.

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (29)$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \quad (30)$$

B. Evaluation Function

Suitable cost evaluation functions are required for PSI-based algorithms to measure the fitness of particles. Due to the high computational efficiency of PSI-based algorithms, fitness evaluation contributes to the majority of their computational time [3]. For path planning, a lower cost/fitness indicates a better solution. The main criteria for evaluating the AUV path are the travel time required to reach the target, the exposure to threats, and the compliance with AUV's physical motion limitations. Since it is almost impossible to achieve all criteria at the same time, a trade-off between these criteria can be established using a weighting scheme with multiple evaluation functions. The fitness of a particle/path X_i can be given by the summation of fitness from multiple evaluation functions F_k for different criteria, with each criterion weighted by a cost factor f_k.

$$F\left(X_i^t\right) = \sum_{k=1}^{K} f_k F_k\left(X_i^t\right), \quad k \in \{1, 2, ..., K\} \quad (31)$$

where k refers to different evaluation functions and K is the total number of functions for the problem.

1) Path Travel Time Cost

The main evaluation function for the path planning problem was to measure the path cost based on its travel time. A given path X_i can be represented in the form of discretised waypoints $P = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$, where P is the output from the B-spline function and m is the number of discretised waypoints. The travel time cost F_1 of a path can be determined using (32).

$$F_1(X_i) = \sum_{j=1}^{m-1} \left[\left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \Big/ \left| V_g \right| \right], \quad j \in \{1, 2, ..., m-1\} \quad (32)$$

where V_g is the resultant ground reference velocity of the AUV, The contribution of current on the AUV can be obtained by projecting the current velocity V_c in the direction of the AUV water reference velocity V_a. Thus, V_g is given as shown in (33).

$$V_g = V_a + \left(V_c \cdot \overrightarrow{p_{i,j}\, p_{i,j+1}} \right) \Big/ \left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \quad (33)$$

2) Threat Cost

Obstacles avoidance of the path planner relied on its threat cost evaluation, which measured the path's exposure to threats. All threats in the problem space were modelled as eclipses in 2D, and as ellipsoids in 3D. The common method to measure the threat cost is by calculating the distance of the discretised waypoints to the centre of threat using (34), and penalising the cost if the distance is smaller than the semi-major axis of threat.

$$d_{threat} = \left\| \overrightarrow{p_{i,j}\, O_{c,h}} \right\|, \quad h \in \{1, 2, ..., H\} \quad (34)$$

where O_c is the threat centre, h refers to different threats and H is the total number of threats in the problem space. However, the accuracy of this method depends on the path fineness, i.e. the number of discretised waypoints on the path. The threat cost can be inaccurate when the distance between two consecutive waypoints is greater than the minor axis of the threat. Therefore, a threat evaluation method based on the intersection between the path and the threats was used. The intersection-based method has fineness-independent accuracy, meaning the computational load can be lowered without affecting the cost accuracy.

Assuming a threat h in 3D problem space with centre $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and semi principal axes $O_{r,h} = (O_{rx}, O_{ry}, O_{rz})$, its parametric equation can be expressed in (35). The equation of a path segment that connects two consecutive waypoints $p_{i,j} = (x_1, y_1, z_1)$ and $p_{i,j+1} = (x_2, y_2, z_2)$ can be written as (36).

$$\left(\frac{x - O_{cx}}{O_{rx}} \right)^2 + \left(\frac{y - O_{cy}}{O_{ry}} \right)^2 + \left(\frac{z - O_{cz}}{O_{rz}} \right)^2 = 1 \quad (35)$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + s \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix} \quad (36)$$

Substituting (36) into (35) can produce the following quadratic equation, which is expressed in term of s.

$$A s^2 + B s + C = 0 \quad (37)$$

$$A = \frac{O_{ry}^2 O_{rz}^2 (x_2 - x_1) + O_{rx}^2 O_{rz}^2 (y_2 - y_1) + O_{rx}^2 O_{ry}^2 (z_2 - z_1)}{O_{rx}^2 O_{ry}^2 O_{rz}^2} \quad (38)$$

$$B = \frac{(O_{cx} - x_1)(x_1 - x_2)}{0.5 O_{rx}^2} + \frac{(O_{cy} - y_1)(y_1 - y_2)}{0.5 O_{ry}^2} + \frac{(O_{cz} - z_1)(z_1 - z_2)}{0.5 O_{rz}^2} \quad (39)$$

$$C = \frac{(O_{cx} - x)^2}{O_{rx}^2} + \frac{(O_{cy} - y)^2}{O_{ry}^2} + \frac{(O_{cz} - z)^2}{O_{rz}^2} - 1 \quad (40)$$

The intersection of the path with the threat can be evaluated by obtaining the discriminant D of (37) according to (41).

$$D = B^2 - 4AC \quad (41)$$

A safety margin was added to the principal axes of all threat regions so that the AUV will not conflict with the threat when $D = 0$. When $D > 0$, the path will conflict with the threat, and the threat cost can be proportional to the length of the segment containing within the threat region. If the path intersects with the threat, the intersection points can be found by solving (37) using (42). The threat cost of a path X_i can then be obtained using (43) .

$$S_1,\ S_2 = \left(-B \pm \sqrt{D}\right)/2A \tag{42}$$

$$F_2(X_i) = \sum_{h=1}^{H}\sum_{j=1}^{m-1} \left\| \overrightarrow{S_1\ S_2} \right\| \Big/ \left[2 \times \max(O_{r,h})\right] \tag{43}$$

3) Physical Motion Limitations

The considerations for physical motion limitations of AUV should include its yaw (turning) and pitch motions. To check the path compliance with the yaw limitation, the turning angle of the path in the *x*-*y* plane was measured and compared against the maximum allowable turning angle $\psi_{\max}$. Considering two consecutive path segments that consist of three waypoints $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j+2}$ (refer to Fig. 1), the turning angle ψ can be obtained from the cosine function as shown in (44).

$$\psi_j = \cos^{-1}\left[\frac{\left(\overrightarrow{p'_{i,j}\ p'_{i,j+1}} \cdot \overrightarrow{p'_{i,j+1}\ p'_{i,j+2}}\right) \Big/ \left\| \overrightarrow{p'_{i,j}\ p'_{i,j+1}} \right\|}{\left\| \overrightarrow{p'_{i,j+1}\ p'_{i,j+2}} \right\|}\right] \tag{44}$$

The cost F_3 for violating the yaw limitation can be obtained from the calculated turning angle as shown in (45).

$$F_3(X_i) = \sum_{j=1}^{m-1} \begin{cases} 0 & , \text{if } |\psi_j| \le \psi_{\max} \\ 1 - \left(180 - |\psi_j|\right) \Big/ \left(180 - \psi_{\max}\right) & , \text{if } |\psi_j| > \psi_{\max} \end{cases} \tag{45}$$

For the pitch motion, the instantaneous pitch angle θ and the change in pitch $\Delta\theta$ of the AUV at any point should not exceed their respective maximum values ($\theta_{\max}$ & $\Delta\theta_{\max}$). Referring to Fig. 1, θ can be determined using a basic tangent function as shown in (46). Next, $\Delta\theta$ can be calculated using (47).

$$\theta_j = \tan^{-1}\left[\left\| \overrightarrow{p_{i,j+2}\ p''_{i,j+2}} \right\| \Big/ \left\| \overrightarrow{p_{i,j+1}\ p''_{i,j+2}} \right\|\right] \tag{46}$$

$$\Delta\theta_j = \theta_{j+1} - \theta_j \tag{47}$$

From the calculated pitch, the cost F_4 for violating $\theta_{\max}$ and the cost F_5 for $\Delta\theta_{\max}$ can be obtained as follows:

$$F_4(X_i) = \sum_{j=1}^{m-1} \begin{cases} 0 & , \text{if } |\theta_j| \le \theta_{\max} \\ 1 - \left(90 - |\theta_j|\right) \Big/ \left(90 - \theta_{\max}\right) & , \text{if } |\theta_j| > \theta_{\max} \end{cases} \tag{48}$$

$$F_5(X_i) = \sum_{j=1}^{m-2} \begin{cases} 0 & , \text{if } |\Delta\theta_j| \le \Delta\theta_{\max} \\ 1 - \left(90 - |\Delta\theta_j|\right) \Big/ \left(90 - \Delta\theta_{\max}\right) & , \text{if } |\Delta\theta_j| > \Delta\theta_{\max} \end{cases} \tag{49}$$

IV. Simulations

A. Simulation Setup

The AUV path planning was conducted in a 1000-run Monte Carlo simulation under 2D scenarios, followed by 3D scenarios. The problem space was a current field that consisted of 50×50 square grids for 2D, and 50×50×50 cube grids for 3D, with each side of the grid equivalent to 1 metre. Non-uniform ocean current and static obstacles of different sizes were present in the problem space. The AUV was required to travel with a pre-set water reference velocity of 1.5m/s. The safety margin used in the threat computation was set to 1 metre, while the angles $\psi_{\max}$, $\theta_{\max}$ and $\Delta\theta_{\max}$ were set to 30°, 45° and 10° respectively. The cost factor for the path travel time, f_1 was set to be 1.0, and the other factors $f_2 - f_5$ were all set to be 0.25. Thus, all costs except the travel time cost had a similar impact on the solutions. In each simulation run, the maximum number of iterations was set to 100. The population size was 150 particles, with each particle consists of 4 path nodes, meaning each particle has 8 dimensions for 2D problems and 12 dimensions for 3D. The algorithm parameters were set to be the values suggested in Section II.

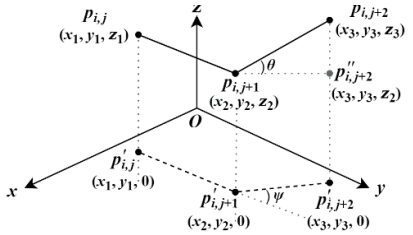


Fig. 1. Yaw angle and pitch angle of a path

B. Simulation Results

The optimal path solutions obtained from the Monte Carlo simulation under 2D and 3D scenarios are shown in Fig. 2 and Fig. 4 respectively. The AUV was required to travel from the starting point (green square) to the target (pink star) without running into the obstacles while trying to take advantage of the favourable current to assist the AUV motion. In 2D (Fig. 2), the blue-coloured zones indicate the favourable current while the red-coloured zones denote the less favourable current. In both domains, the solid sections of the AUV paths indicate that the favourable current has a positive effect on the AUV motion while the dashed sections suggest otherwise. It can be seen that most of the generated paths can follow the favourable zone and avoid the less favourable zone to achieve a shorter travel time.

The performances of the algorithms were compared based on their solution qualities, stabilities, convergence behaviours, and computational requirements; these properties can be evaluated by studying the fitness values of the solutions obtained and the computational time required to obtain the solutions. The fitness value was simply the time required for the AUV to reach the endpoint from the start point by travelling on the path. Thus, a lower fitness value indicates a higher solution quality.

The convergence behaviours of the algorithms under 2D and 3D scenarios are compared in Fig. 3 and Fig. 5. The convergence speed of the algorithm can be given by the minimum number of iterations required for the algorithm to converge at an optimal or sub-optimal solution. It can be observed in the graphs that the convergence speeds of all algorithms significantly decreased when the dimensionality of the problem increased from 2D to 3D. DEPSO and DEQPSO were found to be outperforming other algorithms with similar performance under both scenarios; the two algorithms achieved the fastest convergence and the global convergence with the lowest fitness. The adaptive PSO and IPSO-SQP were able to offer faster and better convergence

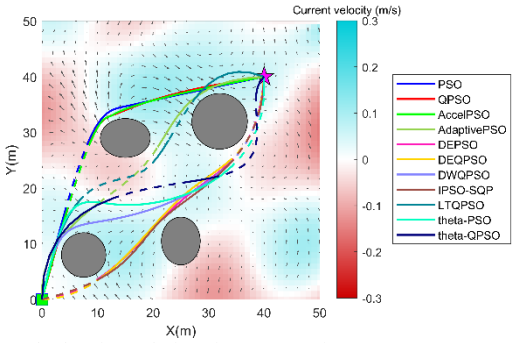


Fig. 2. Path planning solutions in 2D scenario

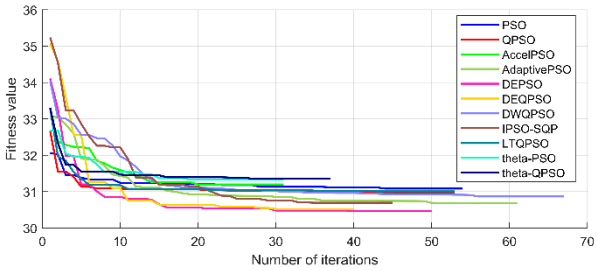


Fig. 3. Convergence curves of fitness values in 2D scenario

than PSO and QPSO under both scenarios. Conversely, θ-PSO and θ-QPSO performed poorly, especially in the 3D scenario. θ-PSO and θ-QPSO were observed to be highly intolerant to local minima when the dimensionality of the problem increased.

The simulation results of 2D and 3D scenarios are graphed in boxplots as shown in Fig. 6, Fig. 7, Fig. 8 and Fig. 9. In the boxplots, the mean of data is represented by the blue plus sign, the median by the red horizontal line, and the blue box on the plot indicates the range of 25th to 75th percentile. The acceptable data range is indicated by the black whisker, and the outliers are represented by red dots. In the fitness value plots, the extreme lowest end of each whisker gives the individual best fitness obtained by each algorithm over the 1000-run simulation, and the green cross sign represents the best known (lowest) fitness value among all algorithms in the simulations. The acceptable data range, percentile range and outliers are indicators for the standard deviations or the stabilities of the performances, while the means and medians give information about the solution qualities and search abilities of the algorithms.

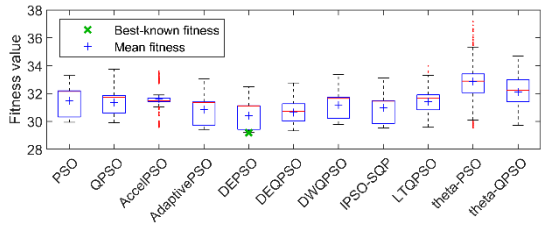


Fig. 6. Boxplot of fitness values in 2D scenario

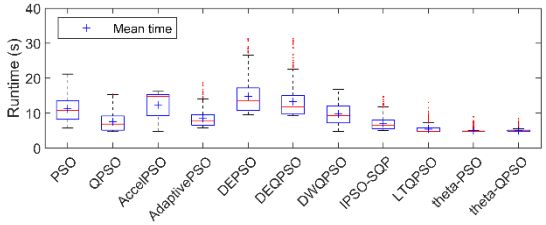


Fig. 7. Boxplot of computational time in 2D scenario

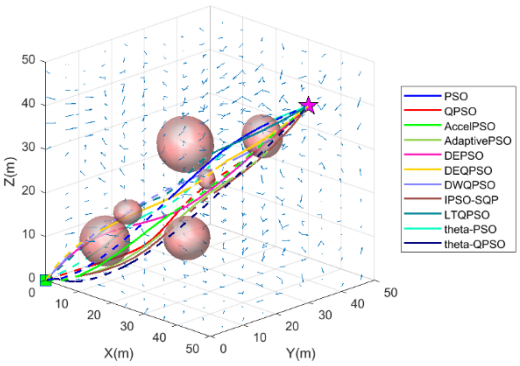


Fig. 4. Path planning solutions in 3D scenario

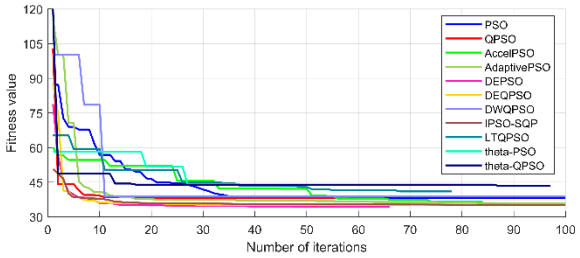


Fig. 5. Convergence curves of fitness values in 3D scenario

It can be seen on the boxplots that DEPSO outperformed other algorithms by achieving the lowest mean fitness value in both 2D and 3D, with its individual best fitness being the best-known fitness value in 2D and the second best-known fitness value in 3D. Following closely the performance of DEPSO, DEQPSO had the second top mean fitness and second best-known fitness in 2D; while for 3D, DEQPSO achieved the fourth top mean fitness and the best-known fitness. It is also worth noting that DEQPSO achieved the lowest standard deviation for the fitness values in 2D, while DEPSO had the lowest standard deviation in 3D. These observations indicated that the hybridization of DE operation into the PSI-based algorithm offered a great improvement to the searching ability and stability of the algorithms. However, as shown in Fig. 7 and Fig. 9, DEPSO and DEQPSO required significantly higher computational time compared to other algorithms, and the increase in computational time was even more obvious when the dimensionality increased to 3D. This is because the greedy selection operator required the fitness values of the particles to

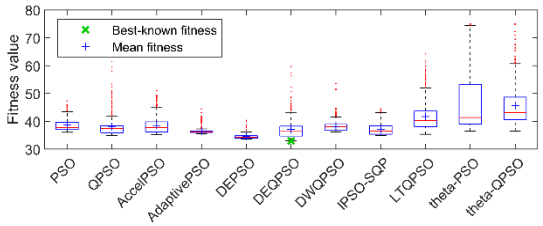


Fig. 8. Boxplot of fitness values in 3D scenario

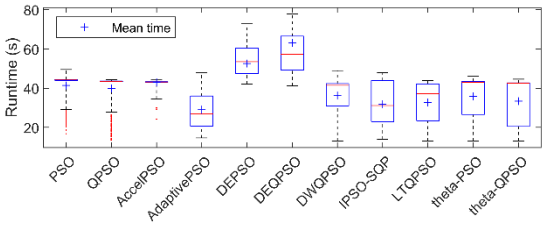


Fig. 9. Boxplot of computational time in 3D scenario

be evaluated twice for comparison purpose, meaning an additional fitness evaluation for every particle in every iteration. Since the fitness evaluation usually contributes to the majority of the computational time, the greedy selection operator drastically increased the computational requirements of the algorithms.

The adaptive PSO and IPSO-SQP were also able to offer excellent performances; both generated higher solution quality than the standard PSO and QPSO. The two algorithms showed a good balance between solution quality and computational time. In 2D, the adaptive PSO was ranked as the third in terms of fitness values, and IPSO-SQP was ranked as the fourth. For 3D, IPSO-SQP had the second top mean fitness, and the adaptive PSO scored third. The two algorithms required less computational time than most algorithms, indicating their high efficiency in solving the path planning problem.

Although DWQPSO achieved a comparable mean fitness, its computational time was significantly higher than the average. The accelerated PSO and LTQPSO did not offer significant improvement over PSO and QPSO in terms of solution quality, despite that LTQPSO required less computation time. θ-PSO and θ-QPSO were found to be performing poorly based on their poorer mean fitness. The extremely low computational time of the three algorithms in 2D indicated that they were prone to be trapped by local minimum. In 3D, the three algorithms had significantly high mean fitness values and high standard deviations, indicating their poor and unstable performance.

V. Conclusion

This paper presents a performance evaluation study to compare various PSI-based algorithms through the application in a novel AUV path planner. Based on the Monte Carlo simulation, both DEPSO and DEQPSO were identified to be outperforming the other algorithms with equivalently excellent performance in terms of solution quality, stability and convergence behaviour, thus proving that the DE hybridization can offer a significant improvement on the particles' searching ability. However, the computational requirement of the DE-hybridized algorithms was observed to be higher due to the greedy selection operator. The adaptive PSO and IPSO-SQP were also found to have excellent performances by achieving a balance between computational requirement and solution quality, with their solution qualities slightly lower than the DE-hybridized algorithms. Most importantly, DEPSO, DEQPSO, the adaptive PSO and IPSO-SQP were proven to be capable of generating high-quality AUV paths.

The future works of this study can be extended in several directions. Firstly, the DE-hybridized algorithms may be improved by modifying the greedy selection operator to reduce their computational requirements, while maintaining their excellent search ability. Next, the possibility of integrating an adaptive mechanism, such as those employed in the adaptive PSO and IPSO-SQP, into the DE-hybridized algorithms can be considered. Lastly, it should be noted that this study considered only static obstacles and non-time-varying currents in the problem space. The potential of these high-performance stochastic algorithms for the application in AUV path planning under realistic environmental conditions should be exploited.

References

[1] T. Xue, R. Li, M. Tokgo, J. Ri, and G. Han, "Trajectory planning for autonomous mobile robot using a hybrid improved QPSO algorithm," *Soft Computing,* vol. 21, no. 9, pp. 2421-2437, 2017.

[2] Z. Zeng, K. Sammut, L. Lian, F. He, A. Lammas, and Y. Tang, "A comparison of optimization techniques for AUV path planning in environments with ocean currents," *Robotics and Autonomous Systems,* vol. 82, pp. 61-72, 2016.

[3] J. Sun, C. H. Lai, and X. J. Wu, *Particle Swarm Optimisation Classical and Quantum Perspectives*. Boca Raton, FL: CRC Press, 2012.

[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. Proceedings of the 6th International Symposium on*, 1995, pp. 39-43: IEEE.

[5] Y. Qin, D. Sun, N. Li, and Q. Ma, "Path planning for mobile robot based on particle swarm optimization," *Robot,* vol. 26, no. 3, pp. 222-225, 2004.

[6] B. Sun, W. D. Chen, and Y. G. Xi, "Particle swarm optimization based global path planning for mobile robots," *Control and Decision,* vol. 20, no. 9, p. 1052, 2005.

[7] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Evolutionary Computation, 2004. CEC2004. Congress on*, 2004, vol. 1, pp. 325-331: IEEE.

[8] Y. Fu, M. Ding, C. Zhou, C. Cai, and Y. Sun, "Path planning for UAV based on quantum-behaved particle swarm optimization," in *Proceedings of SPIE - The International Society for Optical Engineering*, 2009.

[9] Z. B. Shi, Y. Li, X. Wang, T. Wang, and W. M. Li, "Path planning for mobile robot based on quantum-behaved particle swarm optimization," *Harbin Gongye Daxue Xuebao/Journal of Harbin Institute of Technology,* Article vol. 42, no. SUPPL. 2, pp. 33-37, 2010.

[10] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 39, no. 6, pp. 1362-1381, 2009.

[11] B. Song, Z. Wang, L. Zou, L. Xu, and F. E. Alsaadi, "A new approach to smooth global path planning of mobile robots with kinematic constraints," *International Journal of Machine Learning and Cybernetics,* pp. 1-13, 2017.

[12] H. Wang, H. Zhou, and H. Yao, "Research on autonomous planning method based on improved quantum Particle Swarm Optimization for Autonomous Underwater Vehicle," in *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1-7: IEEE.

[13] X. S. Yang, S. Deb, and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," in *International Conference on Networked Digital Technologies*, 2011, pp. 53-66: Springer.

[14] A. Z. Mohamed, S. H. Lee, H. Y. Hsu, and N. Nath, "A faster path planner using accelerated particle swarm optimization," *Artificial Life and Robotics,* vol. 17, no. 2, pp. 233-240, 2012.

[15] W. M. Zhong, S. J. Li, and F. Qian, "θ-PSO: a new strategy of particle swarm optimization," *Journal of Zhejiang University-SCIENCE A,* vol. 9, no. 6, pp. 786-790, 2008.

[16] Y. Fu, M. Ding, and C. Zhou, "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* vol. 42, pp. 511-526, 2012.

[17] W. J. Zhang and X. F. Xie, "DEPSO: hybrid particle swarm with differential evolution operator," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2003, vol. 4, pp. 3816-3821: IEEE.

[18] Y. Fu, M. Ding, C. Zhou, and H. Hu, "Route Planning for Unmanned Aerial Vehicle (UAV) on the Sea Using Hybrid Differential Evolution and Quantum-Behaved Particle Swarm Optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 43, 2013.

[19] H. Modares and M.-B. N. Sistani, "Solving nonlinear optimal control problems using a hybrid IPSO–SQP algorithm," *Engineering Applications of Artificial Intelligence,* vol. 24, no. 3, pp. 476-484, 2011.

[20] T. Taleshian and S. Minagar, "Motion planning for an autonomous underwater vehicle," in *Knowledge-Based Engineering and Innovation (KBEI), 2015 2nd International Conference on*, 2015, pp. 285-290: IEEE.

[21] Q. Qian, M. Tokgo, C. Kim, C. Han, J. Ri, and K. Song, "A hybrid improved quantum-behaved particle swarm optimization algorithm using adaptive coefficients and natural selection method," in *Advanced Computational Intelligence (ICACI), 2015 Seventh International Conference on*, 2015, pp. 312-317: IEEE.

[22] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978.

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-21 (2019) 315–322

IFAC Papers Online CONFERENCE PAPER ARCHIVE

Constrained path planning of autonomous underwater vehicle using selectively-hybridized particle swarm optimization algorithms

Hui Sheng Lim*, Shuangshuang Fan*, Christopher K.H. Chin*, Shuhong Chai*, Neil Bose, Eonjoo Kim***

** National Centre for Maritime Engineering and Hydrodynamics, Australian Maritime College, University of Tasmania, Launceston, TAS, 7250, Australia*
(e-mail: hui.lim; shuangshuang.fan; c.chin; shuhong.chai; eonjoo.kim@utas.edu.au).
*** Department of Ocean and Naval Architectural Engineering, Memorial University of Newfoundland, St. John's, NL A1C 5S7, Canada*
(e-mail: nbose@mun.ca)

Abstract: This paper presents an autonomous underwater vehicle (AUV) path planning scenario as an optimization problem constrained by a combination of hard constraints and soft constraints. The path planner aimed to generate the optimum path that can safely guide an AUV through an ocean environment with a priori known obstacles and non-uniform currents in both 2D and 3D. The path planner used 2 variants of particle swarm optimization (PSO) algorithms, which are the selectively differential evolution (DE)-hybridized quantum PSO (SDEQPSO) and adaptive PSO (SDEAPSO). The performances of the path planners using different constraints were analyzed in a series of extensive Monte Carlo simulations and ANOVA (analysis of variance) procedures based on their respective solution qualities, stabilities and computational efficiencies. Based on the simulation results, the SDEQPSO path planner with the setting of hard constraint for boundary condition and soft constraint for obstacle avoidance was found to be able to generate a smooth and feasible AUV path with higher efficiency than other algorithms, as indicated by its relatively low computational requirement and excellent solution quality.

© 2019, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: path planning, optimization problems, constraints, Monte Carlo simulation, autonomous vehicle

1. INTRODUCTION

To date, numerous efforts have been made in an attempt to enable the operation of AUVs in more dynamic and constrained environments. The exploration of AUVs in highly dynamic regions is challenging and has several technical issues, particularly for the path planning of the AUVs. An optimum AUV path planner should be able to determine a path that safely guides the AUV from a starting position to a destination in an ocean environment, based on either a minimum time or energy cost criterion.

Planning the path for the AUVs is essentially a multimodal and multi-objective optimization problem; numerous techniques have been proposed to solve this problem effectively and efficiently. Nonetheless, developing the algorithms for AUV path planning still faces several intrinsic difficulties, particularly in balancing the computational requirement and the performance of the path planner. Recently, Zeng et al. (2016), and Youakim and Ridao (2018) compared and classified various path planning techniques including artificial potential field (APF), search-based methods, sampling-based methods and optimization methods. APF method (Kruger et al., 2007) is fast and efficient, but very susceptible to local minima. Search heuristic-based planners such as Field D* (Ferguson and Stentz, 2006) and Fast Marching* (FM*) (Petres et al., 2007) are capable of generating optimal and robust path, but their computational efficiencies are limited to less complex and lower dimensional problems. Sampling-based methods like Rapidly-exploring Random Trees (RRT) (Rao and Williams, 2009) and its variants (Hernández et al., 2019) are effective for high-dimensional and highly time-constraint scenarios, but at the cost of the path optimality. Optimization methods such as evolutionary algorithms (Alvarez et al., 2004, Witt and Dunbabin, 2008) show excellent performance in terms of solution optimality. They are effective for high-dimensional complex problems, but their practicality for implementation depends highly on the complexity of their mathematical functions. Among the existing evolutionary algorithms, Zeng et al. (2016) further pointed out that the particle swarm optimization (PSO)-based algorithms are remarkably robust and efficient for solving high-dimensional path planning problem. Lim et al. (2018) compared various PSO-based algorithms for AUV path optimization to identify their strengths and weaknesses. Inspired by these studies, Lim et al. (2020) proposed the selectively differential evolution (DE)-hybridized quantum PSO (SDEQPSO) and adaptive PSO (SDEAPSO), which were developed by hybridizing the PSO algorithm with DE operation based on a selective scheme. They were found to be capable of generating high quality path while maintaining a low computational requirement.

Since the implementation of PSO-based algorithms for path optimization is highly dependent on the mathematical model, it is critical to develop the path planner by formulating the appropriate cost functions and types of constraint. To ensure a smooth, feasible and collision-free path for the AUV, there are many conflicting criteria that need to be considered to achieve

2405-8963 © 2019, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2019.12.326

an optimal control decision. These criteria involve trade-offs between the following objectives: 1) Determine the path with minimum travel time or energy cost; 2) Avoid collision and keep a safe distance with obstacles; 3) Ensure sufficient path control points are placed to generate the path; 4) Ensure the path satisfies the minimum turning radius and the pitch control limitation of the AUV. These criteria render the path planning scenario into a multi-objective optimization problem which can contain two classes of constraints: the hard constraints which must be satisfied by all solutions, and the soft constraints which may or may not be satisfied with different relative weightage (Jiang et al., 1995). The benefit of using a soft constraint over a hard one is that the soft constraint does not need to be satisfied in every iteration, instead, they can be optimized over the iterations; this reduces the solution generation time in every iteration during the optimization (Dariani et al., 2014). If a solution exceeds the soft constraints of the problem, penalty functions with predefined relative weightage can be applied to penalise the fitness of the solution. Choosing the right class of constraint for the path planning problem requires a balance between the computational efforts and the feasibilities of the solutions.

This paper presents a comprehensive comparison between different classes of constraints used for defining the AUV path planning problem, which was solved by using the SDEAPSO and SDEQPSO algorithms. The effect of the types of constraints on the performance of these stochastic PSO-based algorithms were thoroughly analysed. For each test case, the path planning scenario with multiple obstacles and non-uniform current fields was simulated in both 2-dimensional (2D) domain and 3-dimensional (3D) domain. Extensive Monte Carlo simulations were conducted for all test cases and the simulation results were analysed based on their respective solution qualities and stabilities.

The rest of this paper is arranged as follows. In Section 2, an overview of the algorithms used is provided. The formulation of the path planning problem is described in Section 3. Lastly, Section 4 presents the simulation setup, results and discussion.

2. OVERVIEW OF ALGORITHMS

2.1 APSO and QPSO Algorithms

Particle swarm optimization (PSO) is a heuristic population-based optimization algorithm introduced by Eberhart and Kennedy (1995). This algorithm consists of particles that move within a multidimensional search space to search for potential solutions, which are represented by the particles' positions. The particles' velocities are iteratively updated by the particle's own experience (cognitive behaviour) and the entire swarm's experience (social behaviour) to vary the particles' positions. In a standard PSO that consists of N particles with D number of dimensions for solving a cost evaluation function f, the position vector of the i^{th} particle at t^{th} iteration is denoted as:

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ \ldots \ , x_{i,D}^t \right], \quad i \in \{1, 2, ..., N\} \tag{1}$$

Based on its previous best position, *pbest* and global best position in the swarm, *gbest*, the velocity V and the position X of the i^{th} particle at $(t+1)^{\text{th}}$ iteration are updated as follows:

$$V_i^{t+1} = w \cdot V_i^t + C_1 \cdot r_1^t \cdot (pbest_i^{\ t} - X_i^t) + C_2 \cdot r_2^t \cdot (gbest^{\ t} - X_i^t) \tag{2}$$

$$X_i^{\ t+1} = X_i^{\ t} + V_i^{\ t+1} \tag{3}$$

$$pbest_i^{\ t} = \begin{cases} pbest_i^{t-1}, & if\ f(X_i^t) \geq f(pbest_i^{t-1}) \\ X_i^t, & if\ f(X_i^t) < f(pbest_i^{t-1}) \end{cases} \tag{4}$$

$$gbest^{\ t} = \arg\min \left[f(pbest_i^{\ t}) \right] \tag{5}$$

In (2), r_1 and r_2 are random positive numbers that are less than 1.0. C_1 and C_2 are the acceleration coefficients for cognitive and social components respectively, while w is the inertia weight for balancing the particle global exploration and local exploitation to improve the performance. Zhan et al. (2009) proposed the adaptive PSO (APSO), which uses an evolutionary factor f as an indicator representing the particles' evolutionary state to control these equation coefficients. To determine the evolutionary factor f, the mean distance d_i of the i^{th} particle to other particles is calculated using (6). f can be given by (7).

$$d_i = \frac{1}{N-1} \sum_{j=1,\, j \neq i}^{N} \sqrt{\sum_{k=1}^{D} \left(x_i^k - x_j^k \right)^2} \tag{6}$$

$$f = \left(d_g - d_{\min} \right) \Big/ \left(d_{\max} - d_{\min} \right) \quad \in [0,1] \tag{7}$$

where d_g is the mean distance of the global best particle, $d_{\min}$ and $d_{\max}$ are the minimum and maximum of mean distances respectively. f varies from 1 - 0 as the particles move from global exploration to local exploitation phase. w is calculated from f using (8), while C_1 and C_2 can be adapted using (9).

$$w = 1 \Big/ \left(1 + 1.5 e^{-2.6 f} \right) \quad \in [0.4, 0.9] \tag{8}$$

$$\begin{aligned} C_1 &= 0.8 + 2e^{-|f-0.5|} \\ C_2 &= 3.2 - 2e^{-|f-0.5|}, \quad \text{where } C_1 + C_2 = 4 \end{aligned} \tag{9}$$

Inspired by quantum mechanics and PSO, Sun et al. (2004) proposed the QPSO algorithm, which assumes the particles to have quantum behaviour. QPSO algorithm is well known to be an improved version of PSO. In QPSO, the position of the i^{th} particle can be given as:

$$X_i^{t+1} = \begin{cases} \varphi_i^t \cdot pbest_i^{\ t} + \left(1 - \varphi_i^t\right) \cdot gbest^{\ t} \\ \quad + \beta \cdot \left| mbest^{\ t} - X_i^{\ t} \right| \cdot \ln\left(1/u_i^t\right), & if\ u \geq 0.5 \\ \varphi_i^t \cdot pbest_i^{\ t} + \left(1 - \varphi_i^t\right) \cdot gbest^{\ t} \\ \quad - \beta \cdot \left| mbest^{\ t} - X_i^{\ t} \right| \cdot \ln\left(1/u_i^t\right), & if\ u < 0.5 \end{cases} \tag{10}$$

$$mbest^{\ t} = \sum_{i=1}^{N} pbest_i^t \Big/ N \tag{11}$$

where u and φ are random positive numbers that are smaller than 1. β is the contraction-expansion (CE) coefficient, and *mbest* is the mean best position which is defined as the average of personal best positions of all particles as shown in (11). When applying the QPSO algorithm, β is the most critical parameter for controlling the algorithm performance. A linearly decreasing β from $\beta_{\max}$ of 1.0 to $\beta_{\min}$ of 0.5 according to (12) is suggested for most applications (Sun et al., 2012).

$$\beta = \beta_{\max} - (t/t_{\max})(\beta_{\max} - \beta_{\min}) \tag{12}$$

2.2 SDEAPSO and SDEQPSO Algorithms

A selective hybridization of differential evolution (DE) operator with APSO and QPSO was proposed by Lim et al. (2019) to present the SDEAPSO and SDEQPSO algorithms, which were successfully applied to solve an unconstrained AUV path planning problem. Using the selective scheme, these proposed algorithms can apply the DE operation to a selected number of particles only, instead of the entire swarm. The number of particles selected for DE operation, N_S, was controlled by a selective factor S as shown in (13). S was recommended to be 0.3 for AUV path planning problems by Lim et al. (2019) as this setting can help to promote swarm diversity while retaining an adequate group of potentially optimum particles.

$$N_S = N \times S, \quad S \in [0,1] \tag{13}$$

In SDEAPSO and SDEQPSO, the DE operation can initiate by sorting all the particles in the entire swarm according to their personal best positions. Next, a number of selected particles with the best fitness underwent the mutation using (14) to generate the same number of mutated vectors U.

$$U_i^t = gbest^t + \left[(pbest_{i1}^t - pbest_{i2}^t) + (pbest_{i3}^t - pbest_{i4}^t) \right] / 2 \tag{14}$$

where i_1, i_2, i_3 and i_4 are randomly selected particle indices and $i_1 \neq i_2 \neq i_3 \neq i_4 \neq gbest$. The mutated vectors underwent crossover with the personal best positions to generate the same number of trial vectors according to (15).

$$T_i^t = \left[t_{i,1}^t, ..., t_{i,j}^t, ..., t_{i,D}^t \right]$$
$$t_{i,j}^t = \begin{cases} u_{i,j}^t, & if\ r_j \le 0.85 \,||\, j = r \\ pbest_{i,j}^t, & if\ r_j > 0.85 \,||\, j \neq r \end{cases} \tag{15}$$

where r_j is a random number ranging from 0 to 1.0, and r is a random integer ranging from 1 to D. The trial vectors were then subjected to a natural selection operator, in which the same number of particles with the worst fitness were replaced by the trial vectors. Since only the worst particles were replaced in this process, all potentially best solutions never deteriorated. Furthermore, the computational requirement of the algorithms was not significantly affected because the natural selection operator did not involve fitness comparison between the particles, which can require additional particle fitness evaluation in every iteration. The DE operation with natural selection can increase the diversity and the evolutionary rate of the entire swarm by eliminating the least desirable solutions, hence leading to a faster and better global convergence.

The implementation of SDEAPSO and SDEQPSO algorithms in AUV path planning can be conducted as described in the following pseudo code after selecting the appropriate parameters for the algorithm, i.e. the population size N, the number of particle dimensions D and the maximum number of iterations t_{max}.

Step 1. **Input** the algorithm parameters and environmental information of the ocean field.

Step 2. **Initialize** particles with random positions in (1) to represent an initial group of candidate paths. Set *pbest* to be the current particle positions.

Step 3. **While** the stop criteria are not met,

For $t = 1, 2, \ldots, t_{max}$,

SDEAPSO	***SDEQPSO***
Evaluate the cost function $f(X_i^t)$.	Compute *mbest* according to (11).
Update *pbest* and *gbest* according to (4) and (5) respectively.	Evaluate the cost function $f(X_i^t)$.
Update w, C_1 and C_2 according to (8) and (9) respectively.	Update *pbest* and *gbest* according to (4) and (5) respectively.
	Update β according to (12).

For each particle $i = 1, 2, \ldots, N$,

SDEAPSO	***SDEQPSO***
Update particle velocity and position according to (2) and (3) respectively.	Update particle position according to (10).

End

Sort all particles according to the fitness of their personal best positions.

For $k = 1, 2, \ldots, N_S^{th}$ best performing particle,

Mutation: Generate mutated vector U_k^t using (14)

Crossover: Generate trial vector T_k^t using (15).

Natural selection: Replace k^{th} worst-performing particle with trial vector T_k^t.

End

End

Step 4. **Output** gbest that holds the optimal path when the stop criteria are met or when t_{max} is reached.

3. PROBLEM FORMULATION

3.1 Path Formulation

An AUV path planner is required to determine the optimal path among a group of potential paths for the AUV to travel toward a target location. Each potential path can comprise a series of nodes from the start point to the endpoint. Optimizing the coordinates of path nodes can yield the optimal path. The start and end points should not be involved in the optimization because all the potential paths can share the same start and end locations. Each potential path solution for the problem was modelled as an individual particle in the swarm. The swarm population can be denoted by a matrix $X = [X_1, X_2, \ldots, X_N]^T$, where X is the particle's position vector and N is the total number of particles. In this paper, the entries of the position vector represented the polar/spherical coordinates of the path nodes. Assuming a path consists of n+2 nodes including the start and end points, the number of nodes involved in the optimization is n. To record the polar coordinates of n nodes in 2D, the position vector of a particle has 2n dimensions, including n dimensions for radial coordinate r and n dimensions for azimuthal angular coordinate φ. For the spherical coordinates of n nodes in 3D, a particle has 3n dimensions, including additional n dimensions

for polar angular coordinate θ. The position vector of the t^{th} particle at t^{th} iteration for 3D can be given as follows:

$$X_i^t = \left[r_{i,1}^t, r_{i,2}^t, ..., \varphi_{i,n+1}^t, \varphi_{i,n+2}^t, ..., \theta_{i,3n-1}^t, \theta_{i,3n}^t \right] \quad (16)$$

The polar coordinates of a path node in 2D can be converted to Cartesian coordinates using (17), while spherical coordinates in 3D can be converted using (18).

$$\begin{aligned} x_{i,1} &= r_{i,1} \cos \varphi_{i,n+1} \\ y_{i,1} &= r_{i,1} \sin \varphi_{i,n+1} \end{aligned} \quad (17)$$

$$\begin{aligned} x_{i,1} &= r_{i,1} \cos \varphi_{i,n+1} \sin \theta_{i,2n+1} \\ y_{i,1} &= r_{i,1} \sin \varphi_{i,n+1} \sin \theta_{i,2n+1} \\ z_{i,1} &= r_{i,1} \cos \theta_{i,2n+1} \end{aligned} \quad (18)$$

Based on the path nodes including the start and end points, B-spline geometry was used to construct the AUV path. The path nodes can act as the control points for the B-spline curve according to the curve function in (19), which can give output vector $P(u)$ representing a B-spline curve with k+1 order in the form of discretised waypoints. Given the total number of control points is n+2, the total number of piecewise polynomials is one less than the number of control points, which is n+1.

$$P(u) = \sum_{i=0}^{n+1} x_i \, B_{i,k}(u), \qquad i \in \{0,1,2,...,n+1\} \quad (19)$$

where x_i denotes the control points, u is the non-decreasing knot sequence contained in a knot vector $U = [u_0, \ldots, u_i, \ldots, u_{n+k+2}]$, and $B_{i,k}(u)$ represents the piecewise polynomial basis functions of k degree defined by Cox de Boor recursion (De Boor et al., 1978) as follows.

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (20)$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \quad (21)$$

3.2 Path Fitness Evaluation

Suitable fitness evaluation functions are required for PSO-based algorithms to measure the fitness of the particles accurately. Due to the high computational efficiency of PSO-based algorithms, fitness evaluation usually contributes to the majority of computational time (Sun et al., 2012). For path planning, which is a minimization problem, a lower cost/fitness indicates a better solution. In this paper, the main evaluation function was to measure the path fitness based on its time to travel on the path. A given path X_i can be represented in the form of discretised waypoints $P = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$, where P is the output from the B-spline function and m is the number of discretised waypoints. The travel time cost F_1 of a path can be determined using (22).

$$F_1(X_i) = \sum_{j=1}^{m-1} \left[\left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \Big/ \left| V_g \right| \right], \quad j \in \{1,2,...,m-1\} \quad (22)$$

where V_g is the resultant ground reference velocity of the AUV. The contribution of current on the AUV can be obtained by projecting the current velocity V_c in the direction of the water reference velocity V_a. Thus, V_g is given as (23).

$$V_g = V_a + \left(V_c \cdot \overrightarrow{p_{i,j}\, p_{i,j+1}} \right) \Big/ \left\| \overrightarrow{p_{i,j}\, p_{i,j+1}} \right\| \quad (23)$$

3.3 Boundaries and Constraints

Two classes of constraints, namely hard constraint and soft constraint, were used in the AUV path planning problem in order to produce a smooth, feasible and collision-free path that can satisfy the boundaries and the objectives, which included:

- *Obstacle avoidance:* Avoid collision and keep a safe distance from obstacles.
- *Radial boundary:* Ensure sufficient path nodes are placed.
- *Azimuthal boundary:* Ensure the path satisfies the minimum turning radius.
- *Polar boundary:* Ensure the path satisfies the pitch control limitation.

Different combinations of hard and soft constraints were applied to achieve these objectives in this paper. The test cases investigated are summarised in Table 1.

Table 1: Simulation test cases

Objectives	Test cases			
	HBHO	HBSO	SBHO	SBSO
Radial, azimuthal & polar boundaries	Hard	Hard	Soft	Soft
Obstacle avoidance	Hard	Soft	Hard	Soft

The hard constraints must be satisfied by all feasible solutions; while the soft constraints of different relative weightage may or may not be satisfied by the solution. If the hard constraints are violated by a solution, the particular solution will be regenerated. Meanwhile, if the soft constraints are violated, a penalty function with predefined relative weightage will be applied to penalise the fitness of the particle.

To achieve obstacle avoidance, the path's exposure to threats/obstacles was required to be measured regardless of the class of constraint used. All obstacles in the problem space were modelled as eclipses in 2D, and as ellipsoids in 3D. The threat exposure was evaluated based on the intersection between the path and the obstacles. Assuming an obstacle h in 3D problem space with centre $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and semi principal axes $O_{r,h} = (O_{rx}, O_{ry}, O_{rz})$, its parametric equation can be expressed in (24). The equation of a path segment that connects two consecutive waypoints $p_{i,j} = (x_1, y_1, z_1)$ and $p_{i,j+1} = (x_2, y_2, z_2)$ can be written as (25).

$$\left(\frac{x - O_{cx}}{O_{rx}} \right)^2 + \left(\frac{y - O_{cy}}{O_{ry}} \right)^2 + \left(\frac{z - O_{cz}}{O_{rz}} \right)^2 = 1 \quad (24)$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + s \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix} \quad (25)$$

Substituting (25) into (24) can yield the following quadratic equations, which is expressed in term of s.

$$A s^2 + B s + C = 0 \quad (26)$$

$$A=\frac{O_{ry}{}^2O_{rz}{}^2(x_2-x_1)+O_{rx}{}^2O_{rz}{}^2(y_2-y_1)+O_{rx}{}^2O_{ry}{}^2(z_2-z_1)}{O_{rx}{}^2O_{ry}{}^2O_{rz}{}^2} \quad (27)$$

$$B=\frac{(O_{cx}-x_1)(x_1-x_2)}{0.5O_{rx}{}^2}+\frac{(O_{cy}-y_1)(y_1-y_2)}{0.5O_{ry}{}^2}+\frac{(O_{cz}-z_1)(z_1-z_2)}{0.5O_{rz}{}^2} \quad (28)$$

$$C=\frac{(O_{cx}-x)^2}{O_{rx}{}^2}+\frac{(O_{cy}-y)^2}{O_{ry}{}^2}+\frac{(O_{cz}-z)^2}{O_{rz}{}^2}-1 \quad (29)$$

The intersection of the path with the obstacle can be evaluated by obtaining the discriminant D of (26) according to (30).

$$D=B^2-4AC \quad (30)$$

A safe distance was added to the principal axes of all obstacle regions so that the AUV can keep a safe distance from the obstacles and collision will not occur when $D = 0$. If $D > 0$, the collision can be checked by determining (31).

$$s_1,\ s_2=\left(-B\pm\sqrt{D}\right)\big/2A \quad (31)$$

If $s_1<0$ and $s_2>1$, the path will not intersect with the obstacles, i.e. no collision, and hence the hard constraint can be satisfied; otherwise, the path solution will be regenerated. For soft constraint, if a path intersects with the obstacles, the intersection points can be found by solving (26) with (31). The penalty for violating the soft constraint will be proportional to the length of segments containing within the obstacle region as shown in (32). When the soft constraint setting was used for obstacle avoidance, the global best solution of each iteration was still hard-constrained (meaning the iteration will always continue until the global best solution is not penalised), in order to ensure the final solution is collision-free.

$$F_2(X_i)=\sum_{h=1}^{H}\sum_{j=1}^{m-1}\left\|\overrightarrow{S_1\ S_2}\right\|\Big/\left[2\times\max(O_{r,h})\right] \quad (32)$$

To ensure sufficient path nodes were placed to generate the path, each node was constrained to lie within a concentric annulus. The annuli are the regions bounded by every pair of adjacent concentric circles with predefined radii. To achieve this, the radial coordinates of the path nodes were constrained to a lower boundary $R_{\min}$ and an upper boundary $R_{\max}$.

$$\begin{aligned}R_{\min}&=\left[0,r_d,2r_d,\ \ldots\ ,r_{target}\right]\\ R_{\max}&=\left[r_d,2r_d,3r_d,\ \ldots\ ,r_{target}\right]\end{aligned} \quad (33)$$

where r_d is the distance between two concentric circles and r_{target} is the radial coordinate of the target location. The number of path nodes n can be decided by r_d as defined by (34).

$$n=ceil\left[r_{target}/r_d\right] \quad (34)$$

where *ceil* is the rounding function toward positive infinity. The hard constraint can be satisfied if the path solution falls between the boundaries $R_{\min}$ and $R_{\max}$. When using a soft constraint, the following penalty function F_3 was applied.

$$F_3(X_i)=\sum_{j=1}^{n}\begin{cases}0 & ,\ if\ R_{\min,j}\ge r_{i,j}\ge R_{\max,j}\\ R_{\min,j}-r_{i,j} & ,\ if\ r_{i,j}<R_{\min,j}\\ r_{i,j}-R_{\max,j} & ,\ if\ r_{i,j}>R_{\max,j}\end{cases} \quad (35)$$

In order to ensure the minimum turning radius and the pitch limitation were satisfied, the search domains of azimuthal angular coordinate and polar angular coordinate were also constrained within the boundaries φ_{max} and θ_{max}. The path solution can satisfy the hard constraints if $|\varphi_{i,j}| < \varphi_{max}$ and $|\theta_{i,j}| < \theta_{max}$. For soft constraints, the penalty costs followed (36) and (37).

$$F_4(X_i)=\sum_{j=1}^{n}\begin{cases}0 & ,\ if\ |\varphi_{i,j}|\le\varphi_{\max}\\ |\varphi_{i,j}|-\varphi_{\max} & ,\ if\ |\varphi_{i,j}|>\varphi_{\max}\end{cases} \quad (36)$$

$$F_5(X_i)=\sum_{j=1}^{n}\begin{cases}0 & ,\ if\ |\theta_{i,j}|\le\theta_{\max}\\ |\theta_{i,j}|-\theta_{\max} & ,\ if\ |\theta_{i,j}|>\theta_{\max}\end{cases} \quad (37)$$

Using these optimization functions, the test cases in Table 1 were combined with the QPSO, SDEAPSO and SDEQPSO algorithms to solve the path planning problem. The path solutions generated by the path planner were then validated by setting as the reference trajectory for a dynamic model of REMUS 100, which is an under-actuated AUV with a path following controller. Based on Fossen's vectorial representation (Fossen, 1999) and SNAME (Society of Naval Architects and Marine Engineers) standard formulation, the 6 DOF equations of motion for a typical AUV can be modelled as shown in (38) and (39).

$$\dot{\eta}=\begin{bmatrix}R(\eta_2) & 0_{3\times3}\\ 0_{3\times3} & T(\eta_2)\end{bmatrix}\nu \quad (38)$$

$$M\dot{\nu}+C(\nu)\nu+D(\nu)\nu+g(\eta)=\tau \quad (39)$$

where R (η_2) and T (η_2) are the rotation matrices between inertial and body-fixed reference frames for the translational velocities and angular velocities respectively. η in (38) represents the position η_1 and the orientation η_2 of the vehicle with respect to the inertial reference frame, while ν includes the translational velocities ν_1 and the rotational velocities ν_2 of the vehicle with respect to the body-fixed reference frame as described in the vectors in (40).

$$\begin{aligned}\eta&=\begin{bmatrix}\eta_1 & \eta_2\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}x & y & z & \phi & \theta & \psi\end{bmatrix}^{\mathrm{T}},\\ \nu&=\begin{bmatrix}\nu_1 & \nu_2\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}u & v & w & p & q & r\end{bmatrix}^{\mathrm{T}}\end{aligned} \quad (40)$$

In (39), M and $C(\nu)$ describe the inertial and Coriolis matrices (including rigid body and added mass) respectively, while $D(\nu)$ is the hydrodynamics damping matrix, $g(\eta)$ is the hydrostatics restoring forces, and τ describes the control forces from the actuators. This study used the REMUS 100 model derived from equations (38) – (40) by Prestero (2001). The AUV was controlled with a line-of-sight (LOS) guidance controller to follow the trajectory generated by the path planner. The controller used the lookahead-based steering law described by Breivik and Fossen (2009), which can be deemed suitable because of its lower computational requirement and validity for all cross-track errors. The desired yaw angle (heading) ψ_d can be given by the control law in (41). A similar control law was also used for pitch control of the vehicle.

$$\psi_d(e)=\alpha_k+\arctan\left(-K_pe-K_i\int_0^t e(\tau)\,d\tau\right) \quad (41)$$

where α_k is the path-tangential angle, e is the cross-track error, and K_p and K_i are the proportional gain and the integral gain respectively. The integral action in (41) allowed an under-actuated vehicle, such as the REMUS 100, to follow a path regardless of ocean current and non-zero sideslip angles.

4. SIMULATIONS

4.1 Simulation Setup

The AUV path planning was conducted in a 1000-run Monte Carlo simulation under 2D and 3D scenarios. The problem space was a current field that consists of 50×50 square grids for 2D, and 50×50×50 cube grids for 3D, with each side of the grid equivalent to 1 metre. Non-uniform ocean currents and static obstacles of different sizes were present and a priori known in the problem space. The AUV was required to travel with a pre-set water reference velocity of 1.5m/s. The safe distance for obstacle avoidance was set to 1 metre. r_d was set to 20 metres, while the angles $\psi_{\max}$ and $\theta_{\max}$ were set to 60° and 15° respectively. The population size was 150 particles. The algorithm parameters were set to be the values suggested in Section 2.

In addition to all the test cases described in Table 1, test cases with unconstrained path planning problem (uncon.) were also included for comparison purposes. It was discovered that the computational requirement of the SDEAPSO path planner with hard-constrained obstacle avoidance was too high due to the nature of SDEAPSO's position and velocity update equations. Unlike QPSO and SDEQPSO which use the mean best position in their update equations, SDEAPSO has a stronger cognitive component in its equation, making it impossible to satisfy the hard constraint in a single iteration within a reasonable time frame, if the constraint is violated initially. Thus, the HBHO and SBHO cases for SDEAPSO were excluded.

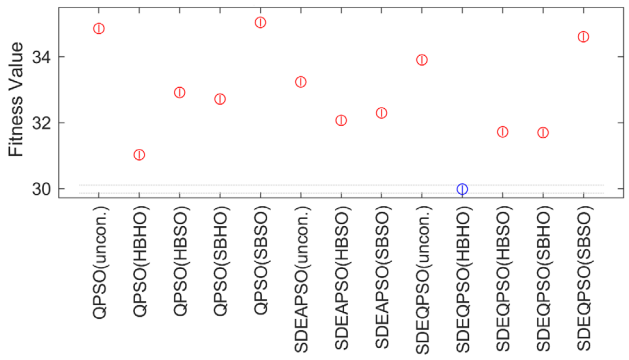


Fig. 1. ANOVA means of fitness values in 2D scenario

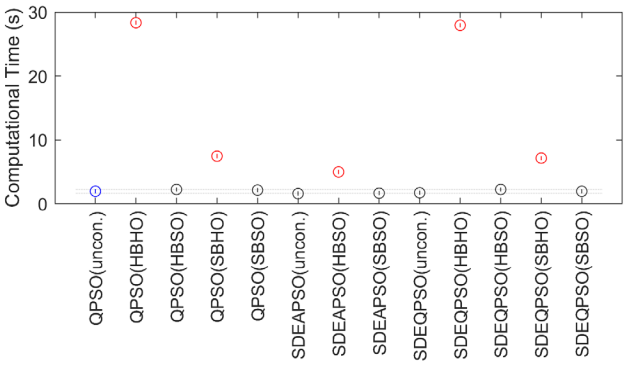


Fig. 2. ANOVA means of computational time in 2D scenario

4.2 Results and Discussion

The performances of the path planners in different test cases were compared based on their solution qualities, stabilities and computational requirements; these properties can be evaluated by studying the fitness values of the solutions obtained and the computational time required to obtain the solutions. The fitness value is simply the time required for the AUV to reach the target by travelling on the path. Thus, a lower fitness value indicates a higher solution quality. To comprehensively compare the test cases and the significance of the differences between their performances, a multiple comparison procedure, ANOVA (analysis of variance), was used in this study with a level of significance of 0.05. This procedure used a 'stepdown' approach, which considered that all but one of the comparisons were less different than the range; such an approach is best suitable for all pairwise comparisons when the confidence intervals are not needed and sample sizes are equal (Sun et al., 2012). The ANOVA results of 2D and 3D scenarios are graphed in Fig. 1, Fig. 2, Fig. 3 and Fig. 4. The best performing results are in blue in the graphs, and those with statistically similar performance to the best performing one are coloured black.

In the 2D scenario, it can be seen that SDEQPSO's HBHO case achieved the best (lowest) fitness value, and this was followed closely by QPSO's HBHO case. Although the HBHO case of SDEAPSO was inadequate for comparison, the HBHO setting was observed to have the best performance in terms of fitness value compared to other settings. However, by comparing the computational time, it was found that the HBHO setting has the highest computational requirement, roughly 10 times of computational time compared to others in 2D. The second-best fitness value was achieved by SDEQPSO's HBSO and SBHO cases. Despite the similar performance, the SBHO case has a much

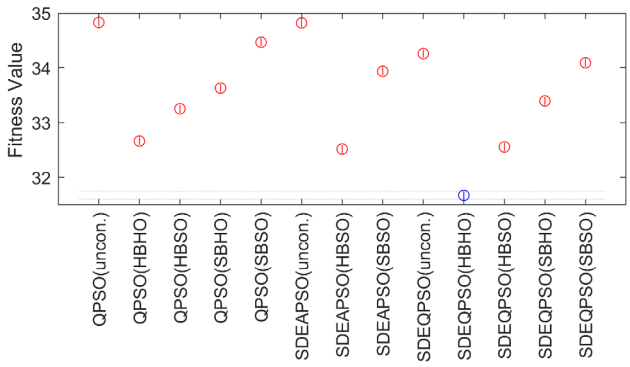


Fig. 3. ANOVA means of fitness values in 3D scenario

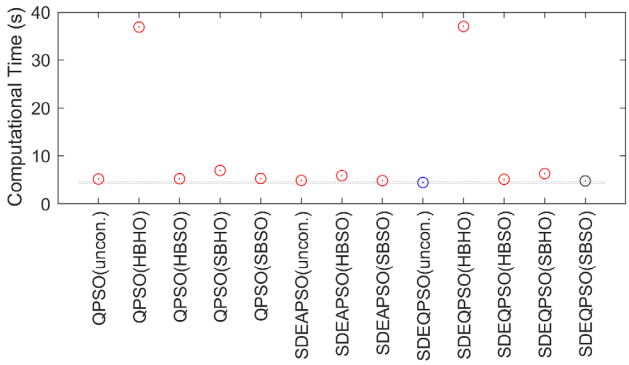


Fig. 4. ANOVA means of computational time in 3D scenario

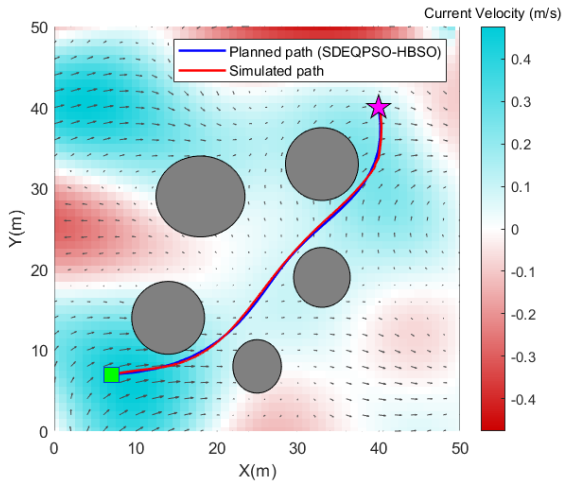


Fig. 5. Validation of path planning solution in 2D scenario

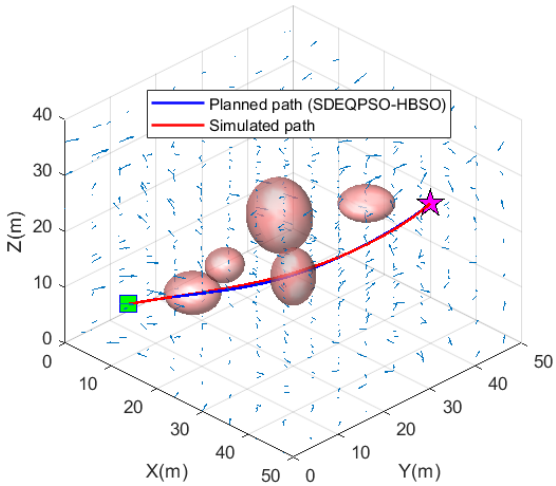


Fig. 8. Validation of path planning solution in 3D scenario

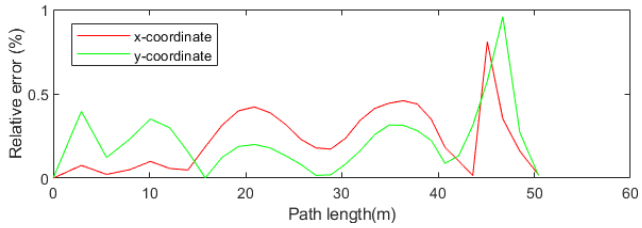


Fig. 6. Relative error of planned and simulated 2D path w.r.t. total path length

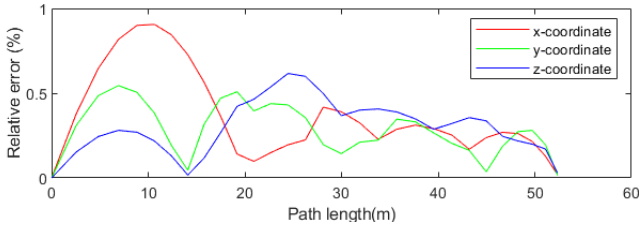


Fig. 9. Relative error of planned and simulated 3D path w.r.t. total path length

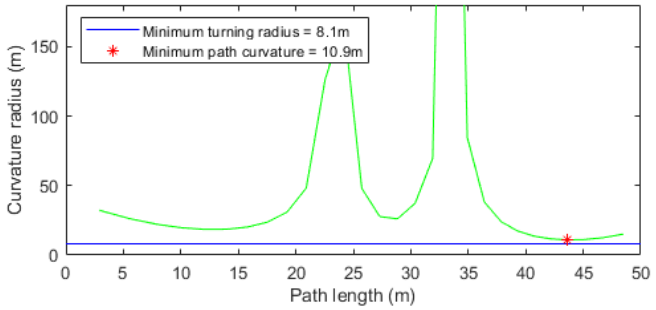


Fig. 7. Curvature radius of the 2D simulated path

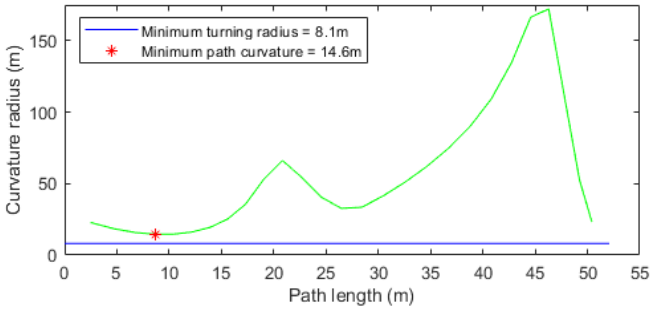


Fig. 10. Curvature radius of the 3D simulated path

higher computational requirement in 2D. Based on this observation, it can be deduced that the hard constraint setting for obstacle avoidance was the main reason for the undesirable increase in computational requirement. When an algorithm-wise comparison was made, it was found that the SDEQPSO had the best overall performance, although SDEAPSO had better performance in the unconstrained case and SBSO case. SDEAPSO was found to have lower performance whenever a hard constraint was involved; this can be explained by its update equation which heavily relied on the cognitive component.

Similar performance trends were observed in the 3D scenario. The HBHO cases achieved the best fitness value but at the cost of a much higher computational requirement. SDEQPSO's HBSO case displayed the second-best fitness value while maintaining a relatively low computational requirement. SDEAPSO was again only able to outperform other algorithms when a hard constraint was not involved. Hence, it can be concluded that the most suitable setting for AUV path planning was the HBSO setting (hard-constrained boundary conditions and soft-constrained obstacle avoidance), which was able to achieve excellent performance in terms of fitness value without high computational requirement.

The 2D and 3D solutions generated by SDEQPSO with HBSO setting were validated by comparing against the simulated paths in Fig. 5 and Fig. 8. The AUV was required to travel from the starting point (green square) to the target (pink star) while keeping a safe distance from obstacles and trying to take advantage of the favourable current to assist the AUV motion. In 2D (Fig. 5), the blue-coloured zones indicate the favourable current while the red-coloured zones denote the less favourable current. Their respective relative errors in each of the x, y and z domains with respect to the total path length are graphed in Fig. 6 and Fig. 9. It can be observed that the simulated paths closely resembled the planned paths, with relative errors of well below 1% for both 2D and 3D scenarios. The feasibility of the path solutions was further checked by comparing against the minimum turning radius of REMUS 100, which has a minimum turning radius of 8.1 metres in the worst-case scenario (Eng et al., 2015). The curvature radius must be higher than the minimum turning radius to satisfy the AUV motion limitation, which can be shown in Fig. 7 and Fig. 10 for the paths in 2D and 3D respectively. Therefore, the simulation results showed that the path solutions generated by the proposed algorithm were smooth and feasible for the path planning application.

CONCLUSIONS

This paper evaluated the performance of an AUV path planner under different types of constraint settings. The SDEQPSO path planner with the setting of hard constraint for boundary condition and soft constraint for obstacle avoidance produced the best performance as shown by its high solution quality and computational efficiency. The path planners with hard constrained obstacle avoidance were found to have a significantly higher computational requirement. Therefore, the soft constraint setting was recommended for obstacle avoidance of the path planner, with the safety and validity of the path guaranteed by having a hard constrained obstacle avoidance on the final solution of each iteration. The proposed path planner successfully generated a feasible and safe path for a REMUS 100 AUV, which was validated through the simulation of the AUV dynamic model.

Although the simulation assumed a priori known environment to represent the minimum capability of path planner, this path planner can be adapted to the realistic operational condition in future work due to the demonstrably high computational efficiency of this stochastic algorithm, which is suitable for solving compute-intensive problems such as path re-planning in a highly dynamic environment. The future extension of this work can be explored by developing a path re-planning algorithm for a priori unknown environment with dynamic obstacles and spatiotemporal ocean currents.

REFERENCES

Alvarez, A., Caiti, A. & Onken, R. 2004. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering,* 29**,** 418-429.

Breivik, M. & Fossen, T. I. 2009. Guidance laws for autonomous underwater vehicles. *Underwater vehicles.* InTech.

Dariani, R., Schmidt, S. & Kasper, R. 2014. Optimization based obstacle avoidance. *International Journal of Computer and Information Engineering***,** 8, 1567-1572.

De Boor, C., De Boor, C., Mathématicien, E.-U., De Boor, C. & De Boor, C. 1978. *A practical guide to splines*, Springer-Verlag New York.

Eberhart, R. & Kennedy, J. 1995. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micro Machine and Human Science. IEEE, 39-43.

Eng, Y., Teo, K. M., Chitre, M. & Ming Ng, K. 2015. *Online System Identification of an Autonomous Underwater Vehicle Via In-Field Experiments*.

Ferguson, D. & Stentz, A. 2006. Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics,* 23**,** 79-101.

Fossen, T. I. 1999. *Guidance and Control of Ocean Vehicles,* Norway, John Wiley & Sons.

Hernández, J. D., Vidal, E., Moll, M., Palomeras, N., Carreras, M. & Kavraki, L. E. 2019. Online motion planning for unexplored underwater environments using autonomous underwater vehicles. *Journal of Field Robotics,* 36**,** 370-396.

Jiang, Y., Kautz, H. & Selman, B. 1995. Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. 1st International Joint Workshop on Artificial Intelligence and Operations Research. 20.

Kruger, D., Stolkin, R., Blum, A. & Briganti, J. 2007. Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 4265-4270.

Lim, H. S., Fan, S., Chin, C. K. & Chai, S. 2018. Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner. 2018 IEEE OES Autonomous Underwater Vehicle Symposium. 1-7.

Lim, H. S., Fan, S., Chin, C. K. H., Chai, S. & Neil, B. 2020. Particle swarm optimization algorithms with selective differential evolution for AUV path planning. *International Journal of Robotics and Automation (IJRA)*, 9(2), 94–112.

Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J. & Lane, D. 2007. Path Planning for Autonomous Underwater Vehicles. *IEEE Transactions on Robotics,* 23**,** 331-341.

Prestero, T. T. J. 2001. *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle.* Massachusetts institute of technology.

Rao, D. & Williams, S. B. 2009. Large-scale path planning for underwater gliders in ocean currents. Australasian Conference on Robotics and Automation (ACRA).

Sun, J., Feng, B. & Xu, W. 2004. Particle swarm optimization with particles having quantum behavior. Proceedings of the 2004 congress on evolutionary computation. IEEE, 325-331.

Sun, J., Lai, C. H. & Wu, X. J. 2012. *Particle Swarm Optimisation Classical and Quantum Perspectives,* Boca Raton, FL, CRC Press.

Witt, J. & Dunbabin, M. 2008. Go with the flow: Optimal AUV path planning in coastal environments. Australian Conference on Robotics and Automation.

Youakim, D. & Ridao, P. 2018. Motion planning survey for autonomous mobile manipulators underwater manipulator case study. *Robotics and Autonomous Systems,* 107**,** 20-44.

Zeng, Z., Sammut, K., Lian, L., He, F., Lammas, A. & Tang, Y. 2016. A comparison of optimization techniques for AUV path planning in environments with ocean currents. *Robotics and Autonomous Systems,* 82**,** 61-72.

Zhan, Z. H., Zhang, J., Li, Y. & Chung, H. S. H. 2009. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* 39**,** 1362-1381.

International Journal of Robotics and Automation (IJRA)
Vol.9, No.2, June 2020, pp. 94~112
ISSN: 2089-4856, DOI: 10.11591/ijra.v9i2.pp94-112

Particle swarm optimization algorithms with selective differential evolution for AUV path planning

Hui Sheng Lim[1], Shuangshuang Fan[2], Christopher K.H. Chin[3], Shuhong Chai[4] and Neil Bose[5]
[1,2,3,4,5]National Centre for Maritime Engineering and Hydrodynamics,
Australian Maritime College, University of Tasmania, Australia
[5]Department of Ocean and Naval Architectural Engineering, Memorial University of Newfoundland, Canada

Article Info

Article history:

Received Jan 13, 2020
Revised Feb 17, 2020
Accepted Mar 4, 2020

Keywords:

Autonomous underwater vehicle
Hybridization
Monte Carlo methods
Particle swarm optimization
Path planning

ABSTRACT

Particle swarm optimization (PSO)-based algorithms are suitable for path planning of an autonomous underwater vehicle (AUV) due to their high computational efficiency. However, such algorithms may produce sub-optimal paths or require a higher computational load to produce an optimal path. This paper proposed a new approach that can improve the ability of PSO-based algorithms to search for the optimal path while maintaining a low computational requirement. By hybridizing with differential evolution (DE), the proposed algorithms can carry out the DE operator selectively to improve the search ability. The algorithms were applied in an offline AUV path planner to generate a Pareto-optimal path that can safely guide the AUV through an environment with a priori known obstacles and time-invariant non-uniform currents. The algorithm performances were benchmarked against other algorithms in an offline path planner because if the proposed algorithms can provide better computational efficiency to demonstrate the minimum capability of a path planner, then they can outperform the tested algorithms in a realistic scenario. Through Monte Carlo simulations and Kruskal-Wallis tests, SDEAPSO (selective DE-hybridized PSO with adaptive factor) and SDEQPSO (selective DE-hybridized quantum-behaved PSO) were found to be capable of generating feasible AUV path with higher efficiency than other algorithms tested, as indicated by their lower computational requirement and excellent path quality.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Hui Sheng Lim,
National Centre for Maritime Engineering and Hydrodynamics,
Australian Maritime College, University of Tasmania,
Launceston, TAS, 7250, Australia.
Email: hui.lim@utas.edu.au

1. INTRODUCTION

AUVs are unmanned underwater vehicles that can be remotely programmed to conduct various missions, ranging from seabed survey, coastal mapping, and environmental monitoring for scientific research purposes, to anti-submarine warfare for defence purposes. To date, numerous efforts have been made in an attempt to enable the operation of AUVs in more dynamic and constrained environments, such as shallow coastal areas, deep ocean regions and regions underneath ice shelves. The operation of AUVs in highly dynamic regions is challenging and it has several technical issues, particularly for the path planning of the AUVs.

Planning the path for an AUV is essentially a multimodal optimization problem; numerous optimization techniques have been proposed to solve this problem effectively and efficiently. Nonetheless, developing the algorithms for AUV path planning still faces several intrinsic difficulties, particularly in

balancing the computational requirements and the performance of the path planner. The high computational requirements for planning the path in a realistic 3D environment may lead to excessive energy drain in an AUV. A common way to keep the computational requirements of path planners feasible is to reduce the problem to a 2D space [1]. This however compromises the performance of the path planner due to the reduced amount of 3D information available for the path planner, such as current fields, bathymetry, and obstacles in the ocean environment. Thus, a computationally efficient algorithm is required for effective AUV path planning in realistic ocean environments.

Recently, Zeng, et al. [2], and Youakim and Ridao [3] compared and classified various path planning techniques including the artificial potential field (APF), search-based methods, sampling-based methods and optimization methods. The APF method [4] is fast and efficient, but very susceptible to local minima. Search heuristic-based planners such as Field D* [5] and Fast Marching* (FM*) [6] are capable of generating optimal and robust paths, but their computational efficiencies are limited to less complex and lower dimensional problems. Sampling-based methods such as rapidly-exploring random trees (RRT) [7] and its variants RRT* [8] are effective for high-dimensional and highly time-constraint scenarios at the cost of the path optimality, and the resultant paths often require further refinement. Meta-heuristic optimization methods such as the evolutionary algorithms [9, 10] showed excellent performance in terms of solution optimality. Evolutionary algorithms are effective for high-dimensional complex problems, but they may converge to local minima within finite time. Among the existing evolutionary algorithms, Zeng, et al. [2] further pointed out that the particle swarm optimization (PSO)-based algorithms are remarkably robust and efficient for solving high-dimensional path planning problems.

PSO algorithm and its most significant variant, the quantum-behaved PSO (QPSO) are extensively used in various optimization problems ever since their emergence in 1995 and 2004 respectively due to their fine search abilities and easy implementations [11]. Some pioneering examples of their applications in path planning can be found in [12-14]. PSO-based path planners are suitable for dynamic environments where online path planning is required because they maintain a large pool of solutions, which is available at any time during the mission. These solutions can serve as the initial solutions whenever path replanning is required, thus significantly improving the computational efficiency. Some successful applications of PSO-based algorithm in online path planning of AUV can be found in [15, 16]. Nonetheless, PSO-based algorithms are susceptible to convergence at local minimum solutions if the time allowed for path planning is limited, which is often the case in real AUV operations.

In recent years, many strategies that modified the PSO and QPSO algorithms have been proposed in order to improve their performances in path planning of various autonomous systems. Each of these variants of the algorithms claimed to have different improvements over the original PSO and QPSO algorithms. To benchmark the PSO and QPSO variants in the application of AUV path planning, a recent comparison study [17] classified and evaluated the algorithms based on their solution qualities, stabilities and computational efficiency. It was concluded from the results of [17] that the hybridization of differential evolution (DE) in PSO and QPSO, which were proposed by [18], were able to produce path planning solution with the highest quality due to their stronger resistance to local minima, but at the cost of higher computational requirements. Moreover, the findings of [17] suggested that having an adaptive mechanism in the evolution of particles in the PSO algorithm can produce solution quality that was second only to DE-hybridized algorithms, but with a relatively low computational requirement; the adaptive PSO (APSO) proposed by [19] was able to generate a path planning solution that achieved a balance between solution quality and computational requirements.

Inspired by the DE hybridization, three algorithms, namely SDEPSO (PSO with selective DE hybridization), SDEAPSO (PSO with adaptive factor and selective DE hybridization), and SDEQPSO (QPSO with selective DE hybridization), are proposed in this paper. These algorithms explored the strengths of DE-hybridized algorithms and minimized their weaknesses in order to improve the algorithm performance. The proposed algorithms were implemented in an offline AUV path planner, and their performances were benchmarked against other meta-heuristic algorithms because if the proposed algorithms can provide better computational efficiency to demonstrate the minimum capability of a path planner, then they can outperform the tested algorithms in a realistic online path planner. The objective of the AUV path planner was defined as finding a near-optimal path that safely guides the AUV from a starting position to a destination based on a minimum time criterion. The path planning scenario with a priori known obstacles and non-uniform current fields was first simulated in a 2-dimensional (2D) domain, followed by the simulation in a 3-dimensional (3D) domain. Extensive Monte Carlo simulations were conducted on all algorithms and the simulation results were analysed based on their respective solution qualities and stabilities.

The rest of this paper is arranged as follows. In Section 2, an overview of the basic PSO, QPSO and their variants, including DEPSO, DEQPSO and APSO is provided. Section 3 describes the novel algorithms proposed in this paper. The formulation of the path planning problem is outlined in Section 4. Section 5 presents the simulation setup, results, and discussions. The generated path solutions were then validated

using an AUV simulator of REMUS 100 in Section 6. Finally, Section 7 concludes the paper along with the future research directions.

2. REVIEW ON PSO AND ITS VARIANTS

This section presents an overview of various particle swarm intelligence-based algorithms used for developing the novel algorithms, which included the basic PSO, basic QPSO and their variants.

2.1. PSO Algorithm

Introduced by Eberhart and Kennedy [20], the PSO algorithm is a heuristic population-based optimization algorithm inspired by the analogues of cognitive abilities and social interaction in animals. The algorithm consists of particles that move within a multidimensional search space to find the potential solutions, which are represented by the particles' positions. The particles' velocities are iteratively updated by the particle's own experience (cognitive behaviour) and the entire swarm's experience (social behaviour) to vary the particles' positions. In a standard PSO algorithm that consists of N particles with D number of dimensions for solving a cost evaluation function f, the position vector of the i^{th} particle at t^{th} iteration can be denoted as:

$$X_i^t = \left[x_{i1}^t, x_{i2}^t, ..., x_{ij}^t, ..., x_{iD}^t \right], \quad i \in \{1, 2, ..., N\} \tag{1}$$

Based on its previous best position *pbest* and global best position in the swarm *gbest*, the velocity V and the position X of the i^{th} particle at $(t+1)^{th}$ iteration are updated by (2) and (3) respectively. *pbest* and *gbest* are determined based on the particle's fitness *f(X)* and its previous best fitness *f(pbest)* as shown in (4) and (5).

$$V_i^{t+1} = w \cdot V_i^t + C_1 \cdot r_1^t \cdot (pbest_i^t - X_i^t) + C_2 \cdot r_2^t \cdot (gbest^t - X_i^t) \tag{2}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{3}$$

$$pbest_i^t = \begin{cases} pbest_i^{t-1}, & if\ f(X_i^t) \geq f(pbest_i^{t-1}) \\ X_i^t, & if\ f(X_i^t) < f(pbest_i^{t-1}) \end{cases} \tag{4}$$

$$gbest^t = \arg\min \left[f(pbest_i^t) \right] \tag{5}$$

In (2), r_1 and r_2 are uniform distributed random positive numbers that are less than 1.0. C_1 and C_2 denote the acceleration coefficients for cognitive and social components respectively; they are both set to 2.0 for most applications [21]. Parameter w is the inertia weight introduced by [22] for balancing the global exploration and local exploitation of the particles. A common strategy is to set the inertia weight at an initial w_{max} value of 0.9, and linearly decreasing to a w_{min} value of 0.4 according to (6) as the iteration progresses [23, 24].

$$w = w_{max} - \frac{t}{t_{max}} (w_{max} - w_{min}) \tag{6}$$

where t_{max} is the maximum number of iterations defined for the algorithm.

To confine the particles within the search space, the particle velocity denoted by V is usually bound to an interval of $[-V_{max}, V_{max}]$, where the maximum velocity V_{max} is recommended to be 10 – 20% of the dynamic range of the variables [24, 25].

2.2. QPSO Algorithm

Inspired by the mechanics of quantum systems and dynamical analysis of the PSO algorithm, Sun, et al. [26] proposed the QPSO algorithm. In QPSO, the position of the i^{th} particle can be updated using the following stochastic equation:

$$X_i^{t+1} = \begin{cases} \varphi_i^t \cdot pbest_i^t + \left(1 - \varphi_i^t\right) \cdot gbest^t + \beta \cdot \left| mbest^t - X_i^t \right| \cdot \ln\left(1/u_i^t\right), & if\ u \geq 0.5 \\ \varphi_i^t \cdot pbest_i^t + \left(1 - \varphi_i^t\right) \cdot gbest^t - \beta \cdot \left| mbest^t - X_i^t \right| \cdot \ln\left(1/u_i^t\right), & if\ u < 0.5 \end{cases} \tag{7}$$

$$mbest^t = \sum_{i=1}^{N} pbest_i^t / N \tag{8}$$

where u is a uniformly distributed random positive number that is less than 1.0, φ is another random number uniformly distributed in the range of 0 – 1.0 , and *mbest* is the mean best position which is defined as the average of personal best positions of all particles in the swarm as shown in (8). β is known as the contraction-expansion (CE) coefficient, which is the most critical parameter for tuning the convergence behaviour of QPSO. As suggested by the empirical study of parameter selection in [11], a linearly decreasing β from a maximum value $\beta_{\max}$ of 1.0 to a minimum value $\beta_{\min}$ of 0.5 according to (9) can be suitable for most applications.

$$\beta = \beta_{\max} - \frac{t}{t_{\max}}\left(\beta_{\max} - \beta_{\min}\right) \tag{9}$$

2.3. DEPSO and DEQPSO Algorithms

One of the most effective methods used for improving the PSO-based algorithm is by hybridization, in which the beneficial features of other optimization techniques are combined with PSO or QPSO algorithm. In [27], the basic PSO was combined with differential evolution (DE), resulting in a hybrid algorithm known as DEPSO. Based on the inspiration from DEPSO, [18] applied a similar hybridization concept in QPSO to propose DEQPSO. In both DEPSO and DEQPSO, the particles undergo the usual position update operations, followed by a successive three-step DE operation, which consists of mutation, crossover and selection as described below.

- Mutation: A mutated donor vector U is first generated using (10).

$$U_i^t = gbest^t + \frac{(pbest_{r1}^t - pbest_{r2}^t) + (pbest_{r3}^t - pbest_{r4}^t)}{2} \tag{10}$$

  where r_1, r_2, r_3 and r_4 are randomly selected particle indices that are mutually different, and different from the current index i and the particle index of global best position, i.e. $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i \neq gbest$.
- Crossover: A trial vector T is generated to increase the diversity, by conducting crossover between the donor vector and personal best position as shown in (11).

$$T_i^t = \left[\tau_{i1}^t, ..., \tau_{ij}^t, ..., \tau_{iD}^t\right]$$
$$\tau_{ij}^t = \begin{cases} u_{ij}^t, & if\ r_j \le CR \,||\, j = r \\ pbest_{ij}^t, & if\ r_j > CR \,||\, j \neq r \end{cases} \tag{11}$$

  where CR is the crossover probability that is suggested to be 0.85, r_j is a uniformly distributed random number ranging from 0 to 1.0, and r is a random positive integer ranging from 1 to the number of particle dimensions D.
- Selection: A greedy selection is used to decide whether the trial vector T should replace the current position X in the (t+1)th iteration. The fitness of T will be evaluated and compared with X. X will only be replaced if T has a better fitness value; otherwise, X will be retained. This means the hybridization of the DE operation will never deteriorate the solution, but only make it better or remain unchanged.

DEPSO and DEQPSO algorithms were applied to solve the path planning problem of an unmanned aerial vehicle (UAV) in [18] and has been proven to be capable of generating significantly higher solution quality than the basic PSO and QPSO algorithms.

2.4. APSO Algorithm

In the basic PSO, the acceleration coefficients C_1 and C_2, and inertia weight w in the update equation are important for maintaining the balance between the global exploration and local exploitation of the particles. Zhan, et al. [19] proposed an adaptive PSO (APSO), in which an evolutionary factor was used as an indicator representing the evolutionary state of the particles to control the equation coefficients adaptively. To determine the evolutionary factor, the mean particle distance d_i of the i^{th} particle to other particles can be calculated using (12). The evolutionary factor f can be computed according to (13).

$$d_i = \frac{1}{N-1}\sum_{j=1,\, j\neq i}^{N}\sqrt{\sum_{k=1}^{D}\left(x_i^k - x_j^k\right)^2} \tag{12}$$

$$f = \left(d_g - d_{\min}\right)\Big/\left(d_{\max} - d_{\min}\right) \quad \in [0,1] \tag{13}$$

where d_g is the mean particle distance of the global best particle, $d_{\min}$ and $d_{\max}$ are the minimum and maximum of the mean particle distances respectively. The inertia weight w can be calculated from evolutionary factor f using (14). The adaptation of the acceleration coefficients C_1 and C_2 can also be achieved using the evolutionary factor as shown in (15).

$$w = 1 \Big/ \left(1 + 1.5e^{-2.6f}\right) \quad \in [0.4, 0.9] \tag{14}$$

$$\begin{aligned} C_1 &= 0.8 + 2e^{-|f-0.5|} \\ C_2 &= 3.2 - 2e^{-|f-0.5|}, \quad \text{where } C_1 + C_2 = 4 \end{aligned} \tag{15}$$

3. METHODOLOGY: SELECTIVE DE HYBRIDIZATION

Although DEPSO and DEQPSO algorithms can generate excellent solution qualities for AUV path planning, the hybridization of DE significantly increases the computational requirements of the algorithm due to the greedy selection operator used in the DE operation [17]. The greedy selection operator requires the fitness of the particles to be evaluated twice for comparison purposes, meaning an additional fitness evaluation for every particle in every iteration. As the fitness evaluation process usually contributes to the majority of the computational time [11], the greedy operator drastically increases the computational requirements of the algorithms. The increase in computational requirements due to the addition of the greedy selection operator can be even more obvious when the complexity and the dimensionality of the problem increase [17]. In order to minimize the downside of DE operator in PSO-based algorithm, a selective hybridization scheme was proposed in this paper to present the following algorithms:

- SDEPSO (PSO with selective DE hybridization)
- SDEAPSO (PSO with adaptive factor and selective DE hybridization)
- SDEQPSO (QPSO with selective DE hybridization)

Using the selective scheme, these proposed algorithms can apply the DE operation to a selected number of particles only, instead of the entire swarm. The number of particles selected for DE operation, N_S, can be controlled by a selection factor S as shown in (16).

$$N_S = N \times S, \quad S \in [0,1] \tag{16}$$

The DE operation in the proposed algorithms was modified by replacing the greedy selection operator with a natural selection operator. The DE operation proposed in this paper can initiate by sorting all the particles in the entire swarm according to their personal best positions. Next, a number of selected particles with the best fitness underwent the mutation and crossover operators, similar to those in DEPSO and DEQPSO, to generate the same number of trial vectors. The trial vectors were then subjected to a natural selection operator, in which the same number of particles with the worst fitness were replaced by the trial vectors.

As only the worst particles can be replaced in this process, all potentially best solutions can never deteriorate. Furthermore, the computational requirements of the algorithms were not significantly affected because the natural selection operator did not involve fitness comparison between the particles, which will require additional particle fitness evaluation in every iteration. The DE operation with natural selection can increase the diversity and the evolutionary rate of the entire swarm by eliminating the least desirable solutions, hence leading to a faster and better global convergence theoretically.

The selective DE hybridization was applied to PSO and QPSO algorithms to develop the SDEPSO and SDEQPSO algorithms in this paper. In addition, another algorithm, namely SDEAPSO, was developed by adding an adaptive mechanism to the control of inertia weight and acceleration coefficients in the PSO algorithm, similarly to the APSO algorithm. The implementation of SDEPSO, SDEAPSO and SDEQPSO algorithms in AUV path planning can be conducted as described in the following pseudocode.

```
Step 1.Input the algorithm parameters and environmental information of the ocean field.
Step 2.Initialize particles with random positions in (1) to represent an initial group of
       candidate paths. Set pbest to be the current particle positions.
Step 3.While the stop criteria are not met,
  Step 3.1 For t = 1, 2, …, tmax,
```

SDEPSO	*SDEAPSO*	*SDEQPSO*
Evaluate the cost function $f(X_i^t)$.	Evaluate the cost function $f(X_i^t)$.	Compute *mbest* according to (8).
Update *pbest* and *gbest* according to (4) and (5) respectively.	Update *pbest* and *gbest* according to (4) and (5) respectively.	Evaluate the cost function $f(X_i^t)$.
Update w according to (6).	Update w, C_1 and C_2 according to (14) and (15) respectively.	Update *pbest* and *gbest* according to (4) and (5) respectively.
		Update β according to (9).

Step 3.2 **For** each particle i = 1, 2, …, N,

SDEPSO	*SDEAPSO*	*SDEQPSO*
Update particle velocity and position according to (2) and (3) respectively.	Update particle velocity and position according to (2) and (3) respectively.	Update particle position according to (7).

End

Step 3.3 Sort all particles according to the fitness of their personal best positions.

Step 3.4 **For** k = 1, 2,…, N_S^{th} best performing particle,

Mutation: Generate mutated vector U_k^t according to (10).

Crossover: Generate trial vector T_k^t according to (11).

Natural selection: Replace k^{th} worst-performing particle with trial vector T_k^t.

End

End

Step 4. **Output** gbest that holds the optimal path when the stop criteria are met or when t_{max} is reached.

3.1. Complexity Analysis

The time complexity of the proposed algorithms can be measured by counting the number of primitive operations in the algorithm. By referring to the pseudocode of the proposed algorithms, the number of operations can be counted as follows:

- In Step 2, initialization contributes to one operation for N times.
- In Step 3.1, cost function evaluation contributes one operation for N times; finding *pbest* requires $N\cdot \log(N)$ operations; finding *gbest* requires $\log(N)$ operations; updating coefficients contributes one operation; SDEQPSO requires N additional operations to calculate *mbest*.
- In Step 3.2, SDEPSO and SDEAPSO perform N loops with 14 operations; SDEQPSO performs N loops with 12 operations.
- Step 3.3 contributes $\log(N)$ operations.
- Step 3.4 performs N_S loops with 8 operations.

Steps 1 – 3.2 are the standard operations in basic PSO, APSO and QPSO, whereas Step 3.3 and 3.4 are the additional operations introduced by the selective DE operator. O-notation was used in this work to denote the asymptotic upper bound of time complexity, which can indicate the computational time of the algorithm in the worst-case scenario. When computing the O-notation, the lower order terms in the number of operations are negligible because their impacts on computational time are relatively insignificant for large input [28]. The highest-order term in the operation is $N\cdot \log(N)$ in Step 3.1, and it performs t_{max} times to check the termination condition. The operations added by the selective DE operator (Step 3.3 and 3.4) are of lower order and do not have a significant impact on the time complexity. Thus, the complexity of the proposed algorithms in linear form is $O(N\cdot \log(N)\cdot t_{max})$, similar to the standard PSO algorithm. PSO-based algorithms have two inner loops when going through the population of N particles, and one outer loop for t_{max} iterations; this renders the time complexity to be $O(N^2\cdot t_{max})$ in the extreme case. The spatial complexity of the algorithms is $O(N^2)$, which depends on the population size.

3.2. Benchmark Functions

Metaheuristic algorithms such as the PSO-based algorithms can be evaluated empirically by comparing their performance in solving a set of objective function problems. In addition to the AUV path planning problem, a number of non-linear continuous function problems was used to study and benchmark the characteristics of the proposed algorithms. According to the "no free lunch" (NFL) theorem [29], the development and evaluation of an algorithm for a specific problem should be based on the benchmark function problems of similar class and properties, because the algorithm performance will not be consistent for every kind of problem. Thus, these benchmark functions were selected based on their resemblances to the properties of the path planning problem. The selected benchmark functions should have the following properties:

- Multimodal with deceptive local minima and one global minimum, because the path planning problem usually consists of multiple suboptimal paths and an optimal path.
- Multi-dimensional, because the dimensionality of the path planning problem is dependent on the number of control waypoints along the path.

Four test functions were chosen for benchmarking in this study. These minimization problem functions, which are commonly used to evaluate the characteristics of optimization algorithms, were found to exhibit the abovementioned properties. The information on the selected benchmark functions is given in Table 1. The dimensions of all functions were set to 20 in this study.

Table 1: Benchmark functions

Notation	Name	Function formulation	Boundary interval	Global minimum
*F*1	Griewank [30]	$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]	$f(x) = 0$, at $x = (0,\ldots,0)$
*F*2	Rastrigin [31]	$f(x)=10d+\sum_{i=1}^{d}\left[x_i^2-10\cos\left(2\pi x_i\right)\right]$	[-5.12, 5.12]	$f(x) = 0$, at $x = (0,\ldots,0)$
*F*3	Ackley [32]	$f(x)=-20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}}-e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}+20+e$	[-32, 32]	$f(x) = 0$, at $x = (0,\ldots,0)$
*F*4	Schwefel [33]	$f(x)=418.9829d-\sum_{i=1}^{d}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500, 500]	$f(x) = 0$, at $x = (420.9687,\ldots, 420.9687)$

3.3. Empirical Study on Parameter Selection

In SDEPSO, SDEAPSO and SDEQPSO, the number of best-performing particles that can undergo the DE operation and the number of worst-performing particles that can be replaced during the natural selection can be determined by the selection factor *S*. Thus, *S* can be manipulated to control the diversity of the population. In order to study the effects of *S* on the algorithm performance, an empirical study was conducted on SDEPSO by using a range of *S*. The selection factor *S* was defined as a positive number that is less than 1.0. Note that when $S = 0$, the algorithm will not be hybridized with DE at all; while $S = 1$ means the DE operation will be conducted on the entire swarm, and the entire swarm will be replaced during the natural selection, meaning all the solutions generated from the PSO operation will be discarded, which can be undesirable. Therefore, the empirical study included *S* values ranging from 0 to 0.9, meaning that 0% – 90% of the particles can undergo the DE operation; the results for $S = 0$ were included for comparison purposes.

Through a 1000-run Monte Carlo simulation with 100 (max) iterations and a population size of 150 particles, the performance of SDEPSO under different *S* settings was evaluated by solving the optimization problems of the benchmark functions and the path planning problem in 2D and 3D scenarios; the formulation of the path planning problem is described in Section 4.

Prior to evaluating the algorithm performance, the Shapiro-Wilk test was performed to examine the normality of the obtained simulation data. The normality test revealed that the data was not normally distributed. Hence, the median was used as the indicator for solution quality. The median of fitness obtained (*Med.*) and the best-known fitness (*Best*) for each setting of *S* were obtained for all problems and tabulated in Table 2. A lower fitness value indicates a higher solution quality and hence a stronger search ability.

Table 2: Empirical study results

Selection factor, *S*	*F*1		*F*2		*F*3		*F*4		2D path planning ($\times 10^2$)		3D Path planning ($\times 10^2$)	
	Med	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*	*Med*	*Best*
0	0.86	0.25	1.28	0.41	0.19	0.06	2.61	1.54	3.07	2.97	3.36	3.30
0.10	0.58	**0.13**	1.22	0.42	**0.15**	0.06	2.22	1.20	3.06	2.99	3.20	3.14
0.20	**0.56**	**0.13**	1.20	0.50	**0.15**	0.07	2.08	0.69	3.01	2.97	3.34	**3.13**
0.30	0.63	0.19	**1.15**	**0.21**	0.17	**0.05**	1.89	0.85	**2.98**	**2.91**	**3.18**	3.14
0.40	0.68	0.34	1.29	0.51	0.23	0.08	1.90	0.81	3.06	2.96	3.30	3.15
0.50	0.66	0.30	1.27	0.40	0.26	0.12	1.71	0.65	3.12	3.02	3.44	3.15
0.60	0.73	0.14	1.41	0.52	0.32	0.11	1.70	0.63	3.05	2.98	3.42	3.18
0.70	0.80	0.34	1.61	0.50	0.47	0.10	1.71	0.60	3.00	2.98	3.33	3.19
0.80	0.87	0.43	1.59	0.84	0.74	0.26	1.51	0.57	3.05	2.97	3.22	3.19
0.90	1.00	0.85	1.77	0.61	1.67	0.43	**1.27**	**0.48**	3.08	2.97	3.35	3.25

The best-performing results for each of the problems are in bold in Table 2. It can be observed from the results that the behaviour of the algorithms varied greatly as *S* increases, and the variations were not consistent for all problems. The best results for the majority of the problems were identified to be in the range of $S = 0.1 – 0.3$, except for problem *F*4. Such results can be explained by the geometry of the Schwefel function *F*4, which has all its local minima and the global minimum spread far apart from one another. Effective optimization of this function requires an algorithm that promotes larger solution diversity (higher *S*) so that it provides a jumping-out ability to prevent trapping in deceptive local minima. This observation complied with the NFL theorem, which suggests that no single algorithm can generate better performance than any other algorithms for every problem. In fact, the improved algorithm performance in one class of problem is not necessarily consistent in all kinds of problems; instead, it is exactly traded with performance in another class of problem [29]. Although all the function problems selected for benchmarking purposes

have similar properties (they are all multimodal and multi-dimensional), the geometry of the problems were different. Therefore, the setting of S should be adjusted accordingly for different optimization problems.

Based on this empirical study, it can be deduced that the optimal setting of S for the majority of the tested problems was in the range of 0.1 – 0.3. More specifically for the path planning problem, the setting of $S = 0.3$ was found to be appropriate and effective.

3.4. Benchmark Study

The benchmark functions were used to evaluate and benchmark the proposed algorithms in this study. Through a 1000-run Monte Carlo simulation with 100 (max) iterations and a population size of 150 particles, the performances of the proposed algorithms in solving the optimization problems of the four benchmark functions were compared with other existing PSO-based algorithms. At each run, the initial particle positions for all problems were randomly generated based on the uniform distribution within the boundary intervals given in Table 3.

As the data was not normally distributed according to the Shapiro-Wilk test, the Kruskal-Wallis test [34], which is a non-parametric ANOVA (analysis of variance), was used with a significance level of 0.05 to rank the algorithm performance based on the solution qualities (fitness obtained). The ranking procedure used the Holm–Bonferroni 'stepdown' approach [35], which is best suited for all pairwise comparisons when the confidence intervals are not needed and sample sizes are equal [11]. The algorithms were given the same rank if they were not statistically different from one another. The medians (*Med.*) of fitness obtained, the ANOVA ranks (*#R*) and the medians of computational time required were tabulated in Table 3. The medians of the top two best-performing results for each problem are in bold. The overall performances of the algorithms can be given by their total ranks, which was calculated from the summation of the ranks of the algorithm for all problems.

Based on the results, it can be seen that there is no single algorithm that can achieve the best results for all problems; this observation agreed with the NFL theory. For the Griewank function (F_1), DEQPSO produced the best result. In fact, APSO, SDEAPSO, QPSO, DEQPSO, and SDEQPSO algorithms were found to be producing satisfactory results, indicating that the adaptive mechanism and quantum behaviour of the particles were beneficial for solving this problem. DEPSO and SDEPSO algorithms produced an equally good performance for the Rastrigin function (F_2). For the Ackley function (F_3), the QPSO-based algorithms, i.e., QPSO, DEQPSO and SDEQPSO produced the best performance, followed by the adaptive PSO-based algorithms, i.e., APSO and SDEAPSO. As far as the Schwefel function (F_4) was concerned, only DEPSO, SDEPSO and SDEAPSO were able to generate satisfactory results, while all the other algorithms showed inferior performances.

The total ranking of the algorithms revealed that DEQPSO achieved better overall performance than other algorithms. The second-best performing algorithms were found to be DEPSO and SDEAPSO. Most importantly, the results for all problems showed that the fully DE-hybridized algorithms, i.e. DEPSO and DEQPSO required significantly higher computational time to obtain the solutions, while the selectively DE-hybridized algorithms were able to maintain a reasonably similar computational requirement as the PSO, QPSO and APSO algorithms.

Table 3: Benchmark study results

Algorithm	*F1*			*F2*			*F3*			*F4*			Total Rank
	Med	*#R*	*T(s)*	*Med*	*#R*	*T(s)*	*Med*	*#R*	*T(s)*	*Med*	*#R*	*T(s)*	
PSO	0.658	8	0.102	1.372	5	0.123	0.453	8	0.104	3.617	5	0.125	26
QPSO	0.089	3	0.160	1.791	6	0.150	0.005	1	0.166	4.555	8	0.187	18
APSO	0.100	4	0.155	1.219	4	0.162	0.041	5	0.177	3.606	5	0.202	18
DEPSO	0.634	6	0.427	**1.140**	1	0.548	0.166	6	0.419	**1.781**	1	0.470	14
DEQPSO	**0.064**	1	0.510	2.092	7	0.502	**0.002**	1	0.490	3.023	4	0.555	13
SDEPSO	0.629	6	0.108	**1.149**	1	0.135	0.172	6	0.177	**1.891**	2	0.199	15
SDEAPSO	0.098	4	0.161	1.196	3	0.157	0.035	4	0.181	2.031	3	0.273	14
SDEQPSO	**0.072**	2	0.177	2.125	7	0.181	**0.002**	1	0.191	3.594	5	0.271	15

4. PROBLEM FORMULATION FOR PATH PLANNING

The AUV path planning problem is formulated in this section. Throughout the formulation, the AUV was assumed to have constant thrust, and hence constant water reference velocity.

4.1. Path Formulation

In this paper, the primary objective of the AUV path planner was to solve a multimodal non-linear optimization problem, in which the optimal path among a group of potential paths for the AUV to travel

towards a target location through the ocean environment was required to be determined. Each potential path of the AUV can comprise a series of nodes along the path from the start point to the target (end) point. Controlling and optimizing the coordinates of the path nodes can yield the optimized path for the AUV. The start point and the endpoint of the path should not be involved in the optimization because all the potential paths share the same start and end locations.

In a PSO-based algorithm, each potential path solution for the problem can be modelled as an individual particle in the swarm population. The swarm population can be denoted by a matrix $X = [X_1, X_2,\ldots, X_N]^T$, where X is the position vector of the particles and N is the number of particles in the swarm. The entries of the position vectors for the particles can represent the coordinates of the path nodes. Assuming every path consists of n+2 nodes including the start point and endpoint, the number of nodes involved in the optimization is n. In order to record the coordinates of n nodes, the entries of the position vector for a particle in 2D problem space has $2n$ dimensions, while a particle in 3D has $3n$ dimensions. Thus, the respective position vectors of the i^{th} particle at t^{th} iteration for 2D and 3D problems are:

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, ..., x_{i,n}^t, x_{i,n+1}^t, ..., x_{i,2n}^t \right], \quad i \in \{1, 2, ..., N\} \tag{17}$$

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, ..., x_{i,n}^t, x_{i,n+1}^t, ..., x_{i,3n}^t \right], \quad i \in \{1, 2, ..., N\} \tag{18}$$

Based on the path nodes including the start and end points, B-spline geometry was used to construct the AUV path. B-splines are parametric curves generated from a series of connected piecewise polynomials [36], which are suitable for modelling the AUV path because of their continuity for smooth path and locality for path alteration without loss of continuity. The path nodes can act as the control points for the B-spline curve according to the following curve function, which gives the output vector $P(u)$ representing a B-spline curve with k+1 order in the form of discretised waypoints. Given the total number of control points is n+2, the total number of piecewise polynomials is one less than the number of control points, which is n+1.

$$P(u) = \sum_{i=0}^{n+1} x_i \, B_{i,k}(u), \qquad i \in \{0, 1, 2, ..., n+1\} \tag{19}$$

where x_i denotes the control points, u is the non-decreasing knot sequence contained in a knot vector $U = [u_0, \ldots, u_i, \ldots, u_{n+k+2}]$, and $B_{i,k}(u)$ represents the piecewise polynomial basis functions of k degree defined by Cox de Boor recursion [37] as follows.

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \tag{21}$$

The continuity of the spline is fully dependent on the basis functions. Hence, it can be noted from (19) that the control points, i.e. path nodes can be adjusted during the path optimization process without affecting the spline continuity.

4.2. Evaluation Functions

When implementing PSO-based algorithms in an optimization problem, it is critical to develop suitable cost evaluation functions to measure the fitness of the particles based on their respective solutions. Due to the high computational efficiency of PSO-based algorithms, the evaluation functions usually contribute to the majority of the computational time [11]. The functions should be developed based on the optimization criteria of the problem. They must closely resemble the physical conditions of the problem space to provide an accurate cost representation model for finding the optimal solution. For path planning, which is a minimization problem, a lower cost/fitness indicates a better solution. The main criteria for evaluating the AUV path were:

- Minimum length or travel time required to reach the target
- Minimum exposure to the threats
- Compliance with physical motion limitations of AUV

As the optimum of all criteria does not necessarily coincide, a trade-off between these criteria can be established using a weighting scheme with multiple evaluation functions, which included the main evaluation function to measure the path length/time cost, a function to measure the threat cost along the path, and

functions to measure the compliance of the path with respect to the AUV motion limitations. Thus, the fitness of a particle/path X_i can be given by a combination of several evaluation functions F_k for different criteria, with each criterion weighted by a cost factor f_k.

$$F\left(X_i^t\right)=\sum_{k=1}^{K} f_k F_k\left(X_i^t\right), \quad k \in \{1,2,...,K\} \tag{22}$$

where k refers to different evaluation functions and K is the total number of functions for the problem.

4.3. Path Travel Time Cost

The main evaluation function for the path planning problem was to measure the path cost based on its length or time to travel on the path. This study focused on finding an optimal path that can take advantage of favourable currents to assist the AUV motion while avoiding less favourable currents to achieve a shorter travel time. For this purpose, a travel-time-based evaluation function was developed in this study.

Based on the previous formulation, a given path X_i can be represented as a series of path nodes or alternatively in the form of discretised waypoints $P = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$, where P is the output from B-spline function and m is the total number of discretised waypoints. The travel time cost F_1 along a path can be determined by finding the sum of discretised time required to travel on each small path segment that connects the consecutive discretised waypoints in P.

$$F_1\left(X_i\right)=\sum_{j=1}^{m-1}\frac{\left\| \overrightarrow{p_{i,j}\ p_{i,j+1}} \right\|}{\left|V_g\right|}, \quad j \in \{1,2,...,m-1\} \tag{23}$$

where V_g is the ground reference velocity of the AUV, which is the resultant AUV velocity under the effect of surrounding ocean current. The contribution of current on the AUV can be obtained by projecting the current velocity V_c in the direction of the AUV water reference velocity V_a, which is essentially the direction of the path vector. Thus, V_g can be given by the sum of V_a and the contribution of V_c as shown in (24).

$$V_g = V_a + \frac{V_c \cdot \overrightarrow{p_{i,j}\ p_{i,j+1}}}{\left\| \overrightarrow{p_{i,j}\ p_{i,j+1}} \right\|} \tag{24}$$

4.4. Threat Cost

The obstacles avoidance ability of the path planner relied on the threat cost evaluation function, in which the exposure of the path to threats/obstacles was measured. All threats in the problem space were modelled as ellipses (or circles if the major axis and minor axis were equal) under 2D conditions, and as ellipsoids (or spheres if all the principal axes were equal) under 3D conditions with their principal axes aligned with the coordinate axes. A threat cost evaluation method based on the intersection between the path and the threats was employed in this study.

Assuming a threat h in 3D problem space with centre $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and semi principal axes $O_{r,h} = (O_{rx}, O_{ry}, O_{rz})$, its parametric equation can be expressed in (25). The parametric equation of a path segment that connects two consecutive waypoints $p_{i,j} = (x_1, y_1, z_1)$ and $p_{i,j+1} = (x_2, y_2, z_2)$ can be written as (26). The cost evaluation in 2D took a similar approach, except that the dimension reduction in 2D reduced the number of variables and hence simplified the computation.

$$\left(\frac{x-O_{cx}}{O_{rx}}\right)^2+\left(\frac{y-O_{cy}}{O_{ry}}\right)^2+\left(\frac{z-O_{cz}}{O_{rz}}\right)^2=1 \tag{25}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}=\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}+s\begin{pmatrix} x_2-x_1 \\ y_2-y_1 \\ z_2-z_1 \end{pmatrix} \tag{26}$$

Substituting (26) into (25) yields the following equations, which are expressed in terms of s. The intersection of the path with the threat can be evaluated by obtaining the discriminant ξ of (27) according to (31).

$$As^2+Bs+C=0 \tag{27}$$

$$A = \frac{O_{ry}{}^2 O_{rz}{}^2 (x_2 - x_1) + O_{rx}{}^2 O_{rz}{}^2 (y_2 - y_1) + O_{rx}{}^2 O_{ry}{}^2 (z_2 - z_1)}{O_{rx}{}^2 O_{ry}{}^2 O_{rz}{}^2} \tag{28}$$

$$B = \frac{(O_{cx} - x_1)(x_1 - x_2)}{0.5 O_{rx}{}^2} + \frac{(O_{cy} - y_1)(y_1 - y_2)}{0.5 O_{ry}{}^2} + \frac{(O_{cz} - z_1)(z_1 - z_2)}{0.5 O_{rz}{}^2} \tag{29}$$

$$C = \frac{(O_{cx} - x)^2}{O_{rx}{}^2} + \frac{(O_{cy} - y)^2}{O_{ry}{}^2} + \frac{(O_{cz} - z)^2}{O_{rz}{}^2} - 1 \tag{30}$$

$$\xi = B^2 - 4AC \tag{31}$$

A safety margin was added to the principal axes of all threat regions so that the AUV did not conflict with the threat when $\xi = 0$, i.e., the path was tangent to the threat region. When $\xi > 0$, the path conflicted with the threat if the roots s_1 and s_2 given by (32) were within the range of $0 \leq s_1, s_2 \leq 1$.

$$s_1,\ s_2 = \frac{-B \pm \sqrt{\xi}}{2A} \tag{32}$$

If the path conflicted with the threat, the threat cost was proportional to the length of the path segment contained in the threat region as given in (33). The intersection points, S_1 and S_2 can be determined by solving (27) using the obtained s_1 and s_2 and substituting them back into (26).

$$F_2\left(X_i\right) = -\sum_{h=1}^{H} \sum_{j=1}^{m-1} \frac{\left\| \overrightarrow{S_1\ S_2} \right\|}{2 \times \max\left(O_{r,h}\right)} \tag{33}$$

4.5. Physical Motion Limitations

The considerations for physical motion limitations of an AUV should include its yaw (turning) and pitch motions at a given forward speed. Evaluation functions were developed to check the compliance of the path with respect to these limitations and to penalise the cost if any of the limitations were violated. To check the path compliance with the yaw limitation, the turning angle of the path in the *x-y* plane was measured and compared against the maximum allowable turning angle $\psi_{\max}$. Considering two consecutive path segments that consist of three waypoints $p_{i,j}$, $p_{i,j+1}$ and $p_{i,j+2}$ (refer to Figure 1), the turning angle ψ can be obtained from the cosine function as shown in (34). The first part of the function is the scalar projection of the second path segment on the first segment in the *x-y* plane, while the second part is the length of the second path segment in the *x-y* plane.

$$\psi_j = \cos^{-1}\left[\frac{\overrightarrow{p'_{i,j}\ p'_{i,j+1}} \cdot \overrightarrow{p'_{i,j+1}\ p'_{i,j+2}}}{\left\| \overrightarrow{p'_{i,j}\ p'_{i,j+1}} \right\|} \times \frac{1}{\left\| \overrightarrow{p'_{i,j+1}\ p'_{i,j+2}} \right\|} \right] \tag{34}$$

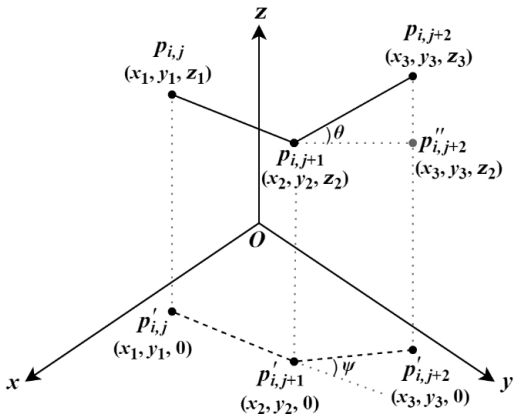


Figure 1: Yaw angle and pitch angle of a path

The cost F_3 for violating the yaw limitation was obtained from the calculated turning angle as shown in (35).

$$F_3(X_i) = -\sum_{j=1}^{m-1} F_3(p_{i,j})$$
$$F_3(p_{i,j}) = \begin{cases} 0 & , \ if \ |\psi_j| \le \psi_{\max} \\ \left(180 - |\psi_j|\right) \big/ \left(180 - \psi_{\max}\right) & , \ if \ |\psi_j| > \psi_{\max} \end{cases} \quad (35)$$

For the pitch motion, the instantaneous pitch angle θ and the change in pitch $\Delta\theta$ of the AUV at any point should not exceed their respective maximum values ($\theta_{\max}$ & $\Delta\theta_{\max}$). Referring to Figure 1, θ can be determined using the basic tangent function as shown in (36). Next, $\Delta\theta$ can be calculated using (37).

$$\theta_j = \tan^{-1}\left[\frac{\left\| \overrightarrow{p_{i,j+2}\ p''_{i,j+2}} \right\|}{\left\| \overrightarrow{p_{i,j+1}\ p''_{i,j+2}} \right\|}\right] \quad (36)$$

$$\Delta\theta_j = \theta_{j+1} - \theta_j \quad (37)$$

From the calculated pitch, the cost F_4 for violating $\theta_{\max}$ and the cost F_5 for $\Delta\theta_{\max}$ can be obtained as shown in (38) and (39) respectively.

$$F_4(X_i) = -\sum_{j=1}^{m-1} F_4(p_{i,j})$$
$$F_4(p_{i,j}) = \begin{cases} 0 & , \ if \ |\theta_j| \le \theta_{\max} \\ \left(90 - |\theta_j|\right) \big/ \left(90 - \theta_{\max}\right) & , \ if \ |\theta_j| > \theta_{\max} \end{cases} \quad (38)$$

$$F_5(X_i) = -\sum_{j=1}^{m-2} F_5(p_{i,j})$$
$$F_5(p_{i,j}) = \begin{cases} 0 & , \ if \ |\Delta\theta_j| \le \Delta\theta_{\max} \\ \left(90 - |\Delta\theta_j|\right) \big/ \left(90 - \Delta\theta_{\max}\right) & , \ if \ |\Delta\theta_j| > \Delta\theta_{\max} \end{cases} \quad (39)$$

5. SIMULATIONS

The performance of the proposed algorithms was evaluated in the AUV path planning problem under different scenarios in this section.

5.1. Simulation Setup

The path planning of the AUV was conducted in a 1000-run basis Monte Carlo simulation under a 2D scenario, followed by a 3D scenario. The machine used has an Intel Core i5-6300U CPU @ 2.4GHz with 8GB RAM. The problem spaces of the simulations were assumed to be a current field that consists of 500×500 square grids for 2D, and 500×500×500 cube grids for 3D, with each side of the grid equivalent to 1 metre. Non-uniform ocean current and static obstacles of different sizes were present in the problem space. The current field was generated based on the data obtained from the field experiment conducted at Beauty Point, Tasmania, Australia.

The AUV was required to travel from a starting point to a target with a pre-set water reference velocity of 1.5m/s. Based on the properties of REMUS 100 AUV, the safety margin used in the threat computation was set to 1 metre, while the angles $\psi_{\max}$, $\theta_{\max}$ and $\Delta\theta_{\max}$ were set to 30°, 45° and 10° respectively. The cost factor for the path travel time f_1 was set to be 1.0, and other cost factors $f_2 - f_5$ were all set to be 0.25 so that all costs except the travel time cost can have a similar impact on the solutions. Hence, when the path solution was not violating the threat exposure (f_2) and the physical motion limitations ($f_3 - f_5$), the fitness value of the solution can directly represent the time required for the AUV to travel on the corresponding path.

In each simulation run, the maximum number of iterations for the algorithm was set to 100 with a pre-defined stopping threshold. This means the algorithm can iterate up to a maximum number of 100 but was stopped whenever the solution difference between iterations was less than the pre-set threshold. The

population size of all algorithms was set to 150 particles, with each particle consisting of 5 path nodes, meaning each particle had 10 dimensions for the 2D problem and 15 dimensions for the 3D problem. All algorithm parameters were set to be their respective suggested values as discussed in Section 2. For comparison purposes, another path planning technique, RRT* and other metaheuristic algorithms, including ant colony optimization (ACO) [38], firefly algorithm (FA) [16], differential evolution (DE) and genetic algorithm (GA) [9], were also tested in this study.

5.2. Simulation Results

The performances of the algorithms were compared based on the following properties: solution qualities, stabilities, convergence behaviours, and computational requirements. These properties can be evaluated by studying the fitness values of the solutions obtained and the computational time required to obtain the solutions. The fitness value of a solution was simply the time required (cost) for the AUV to reach the endpoint from the starting point by travelling on the path corresponding to the solution. Therefore, a lower fitness value can indicate a higher solution quality and hence a stronger search ability.

The Monte Carlo simulation results of the 2D and 3D scenarios are graphed and compared in boxplots as shown in Figure 2 and Figure 3. The data was not normally distributed based on the Shapiro-Wilk normality test. In the boxplots, the medians of the data are represented by the red horizontal line; the blue boxes indicate the range of 25th to 75th percentile; the black whiskers indicate the acceptable data range. For the boxplots of fitness values, the extreme lowest end of each whisker gives the individual best fitness obtained by each algorithm over the 1000-run simulation, and the green cross sign represents the best known (lowest) fitness value among all algorithms in the simulations. The acceptable data ranges and percentile ranges are indicators for the stabilities of the algorithm performances, while the medians give information about the solution qualities and search abilities of the algorithms.

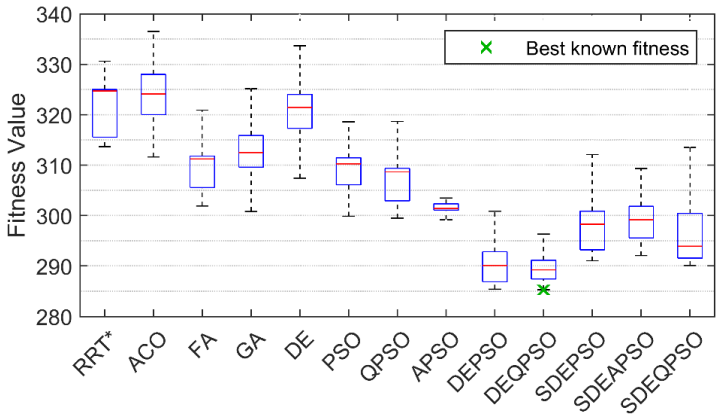


Figure 2: Boxplot of fitness values in 2D scenario

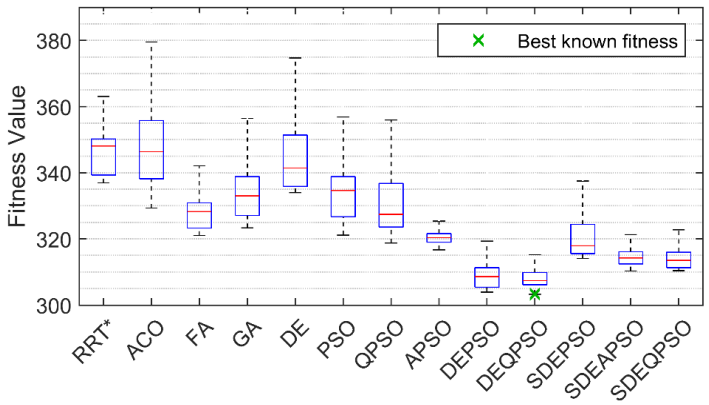


Figure 3: Boxplot of fitness values in 3D scenario

The Kruskal-Wallis ANOVA procedure with a significance level of 0.05 was used to rank the solution qualities (fitness values) based on the Holm–Bonferroni step-down method. The algorithms were given the same rank if they are not statistically different from one another. Detailed results of the path planning simulation, including the median of fitness obtained (*Med.*), the best-known fitness (*Best*), the interquartile range (*IQR*), the ANOVA rank (*#R*), the median of computational time (*T*) and the total ranks, are tabulated in Table 4. The total ranks are calculated from the summation of the ranks for the 2D and 3D scenarios. The ranking of the algorithms did not consider the impact of computational time.

Based on Figure 2, Figure 3 and Table 4, almost all the PSO-based algorithms have better solution quality than RRT* and other metaheuristic algorithms, with the exception of standard PSO being outperformed by FA. Despite having lower solution quality, RRT* has the shortest computational time in both 2D and 3D scenarios. It can also be seen that all variants of PSO and QPSO produced better solution qualities than the standard PSO and QPSO. DEPSO and DEQPSO outperformed all other algorithms by achieving the lowest medians for fitness value in both 2D and 3D. The total ranks of DEPSO and DEQPSO suggest that the two fully DE-hybridized algorithms were able to produce the top two best solution qualities for the path planning problem. However, the computational time of DEPSO and DEQPSO were significantly higher than all the other algorithms due to the high computational requirements of the greedy selection operator.

Table 4: Path planning simulation results

Algorithm	2D					3D					Total Rank
	Med. ($\times 10^2$)	*Best* ($\times 10^2$)	*IQR*	*#R*	*T(s)*	*Med.* ($\times 10^2$)	*Best* ($\times 10^2$)	*IQR*	*#R*	*T(s)*	
RRT*	3.25	3.14	9.4	11	4.8	3.48	3.37	10.9	11	14.3	22
ACO	3.24	3.12	8.0	13	9.4	3.46	3.29	17.6	11	41.3	24
FA	3.11	3.02	6.2	8	9.2	3.28	3.21	7.7	7	41.2	15
GA	3.13	2.98	6.3	10	12.3	3.33	3.23	11.7	9	48.3	19
DE	3.21	3.05	6.7	11	12.8	3.41	3.34	15.5	11	53.6	22
PSO	3.10	3.00	5.4	8	10.7	3.35	3.21	12.1	9	34.6	17
QPSO	3.09	3.00	6.4	7	9.9	3.27	3.19	13.2	7	30.9	14
APSO	3.01	2.92	1.3	5	10.8	3.20	3.17	2.6	5	37.7	10
DEPSO	2.90	2.85	5.9	1	22.4	3.09	3.04	5.9	1	69.0	2
DEQPSO	2.89	2.85	3.7	1	20.8	3.07	3.03	3.7	1	76.7	2
SDEPSO	2.98	2.91	7.7	6	12.8	3.18	3.14	8.8	5	35.7	11
SDEAPSO	2.99	2.92	6.2	3	14.9	3.14	3.10	3.7	4	38.8	7
SDEQPSO	2.94	2.90	7.8	3	13.7	3.13	3.10	4.7	3	37.6	6

The solution qualities of SDEAPSO and SDEQPSO were second to the fully DE-hybridized algorithms; they were ranked similarly in 2D based on the ANOVA ranking. APSO had better solution quality than SDEPSO in 2D. It is worth noting that APSO had the lowest interquartile range is both 2D and 3D, indicating the highest stability among all the algorithms. In the 3D scenario, SDEQPSO was ranked slightly higher than SDEAPSO, while SDEPSO was ranked similar to APSO. The total ranks of the overall performance in both 2D and 3D revealed that SDEQPSO and SDEAPSO were ranked as the third and the fourth respectively. More importantly, the computational times of the two selectively DE-hybridized algorithms were significantly lower than the fully DE-hybridized algorithms and very close to other PSO-based algorithms. These indicate the higher computational efficiency of SDEQPSO and SDEAPSO in solving the path planning problem because they were able to produce solution quality that was very close to DEPSO and DEQPSO while having a significantly lower computational requirement. In terms of problem size, the computational time required by the path planner was considered short, particularly in comparison to the computational time required for estimating the ocean environment based on the AUV sensory measurements.

6. VEHICLE PATH VALIDATION

For validation purposes, the path solutions generated by the AUV path planner were used as a reference trajectory for a dynamic model of REMUS 100. This section briefly explains the dynamic model and the path following controller used.

6.1. Dynamic Model

Based on Fossen's vectorial representation [39] and SNAME (Society of Naval Architects and Marine Engineers) standard formulation, the 6 DOF equation of motion for a typical AUV can be modelled as shown in (40) and (41).

$$\dot{\eta} = \begin{bmatrix} R(\eta_2) & 0_{3\times3} \\ 0_{3\times3} & T(\eta_2) \end{bmatrix} \nu \tag{40}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \tag{41}$$

where $R(\eta_2)$ and $T(\eta_2)$ are the rotation matrices between inertial and body-fixed reference frames for the translational velocities and angular velocities respectively. η includes the position η_1 and the orientation η_2 of the vehicle with respect to the inertial reference frame, while the derivative of η in (40) represents its rate of change. ν includes the translational velocities ν_1 and the rotational velocities ν_2 of the vehicle with respect to the body-fixed reference frame as described in the vectors in (42).

$$\begin{aligned} \eta &= \begin{bmatrix} \eta_1 & \eta_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^{\mathrm{T}}, \\ \nu &= \begin{bmatrix} \nu_1 & \nu_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{42}$$

In (41), M and $C(\nu)$ describe the inertial and Coriolis matrices (including rigid body and added mass) respectively, $D(\nu)$ is the hydrodynamics damping matrix, $g(\eta)$ is the hydrostatics restoring forces, and τ describes the control forces from the actuators. This study used the REMUS 100 model derived from (40)–(42) by [40]. The hydrodynamics coefficients calculated in [40] were used in the vehicle model.

6.2. Path Following Controller

The path following controller of the AUV model used the integral line-of-sight (iLOS) guidance law to set the yaw and pitch angles for following the trajectory generated by the path planner. The iLOS guidance law described by [41] allowed the AUV to shape the convergence towards the path in the presence of ocean current and environmental disturbance. The desired iLOS yaw angle (heading) ψ_d and pitch angle θ_d can be given as follows:

$$\begin{aligned} \psi_d(e) &\triangleq \arctan\left(\frac{e + K_{i,y} e_{\mathrm{int}}}{\Delta_y}\right), \qquad K_{i,y},\ \Delta_y > 0 \\ \dot{e}_{\mathrm{int}} &= \frac{\Delta_y e}{\left(e + K_{i,y} e_{\mathrm{int}}\right)^2 + \Delta_y^2} \end{aligned} \tag{43}$$

$$\begin{aligned} \theta_d(h) &\triangleq \arctan\left(\frac{h + K_{i,z} h_{\mathrm{int}}}{\Delta_z}\right), \qquad K_{i,z},\ \Delta_z > 0 \\ \dot{h}_{\mathrm{int}} &= \frac{\Delta_z h}{\left(h + K_{i,z} h_{\mathrm{int}}\right)^2 + \Delta_z^2} \end{aligned} \tag{44}$$

where e is the cross-track error, h is the vertical-track error, $K_{i,y}$ and $K_{i,z}$ are the integral gains, and Δ_y and Δ_z represent the look-ahead distances for iLOS heading and pitch respectively. The integral terms of cross-track error e_{int} and vertical-track error h_{int} can produce non-zero ψ_d and θ_d even when the AUV is on the planned path, allowing the vehicle to counteract any effects of ocean current with the necessary side-slip and pitch angles. The rates of integral terms $\dot{e}_{int}$ and $\dot{h}_{int}$ can reduce the integral action with large cross-track and vertical-track errors (i.e. vehicle is far from the planned path), in order to minimize the risk of integrator wind-up.

6.3. Validation Results

The feasibility of the path solutions was first checked against the motion limitation of REMUS 100, which has a minimum turning radius of 8.1 metres in the worst-case scenario [42]. The curvature radius of a feasible path must be higher than the minimum turning radius. The paths generated by SDEQPSO satisfied the AUV motion limitation as shown in Figure 4.

Next, the 2D and 3D solutions generated by SDEQPSO were validated by checking against the simulated paths in Figure 5. The AUV was required to travel from the starting point (green square) to the target (pink star) without running into obstacles while trying to take advantage of the favourable current to assist the AUV motion. In the 2D results, the blue-coloured regions indicate the favourable current while the red-coloured regions denote the less favourable current. In both results, the solid sections of the planned paths indicate that the favourable ocean current has a positive effect on the AUV motion while the dashed sections suggest otherwise. It can be observed that the paths were able to follow the favourable current and avoid the less favourable current to achieve a shorter travel time. The simulated paths closely resembled the planned paths in both scenarios.

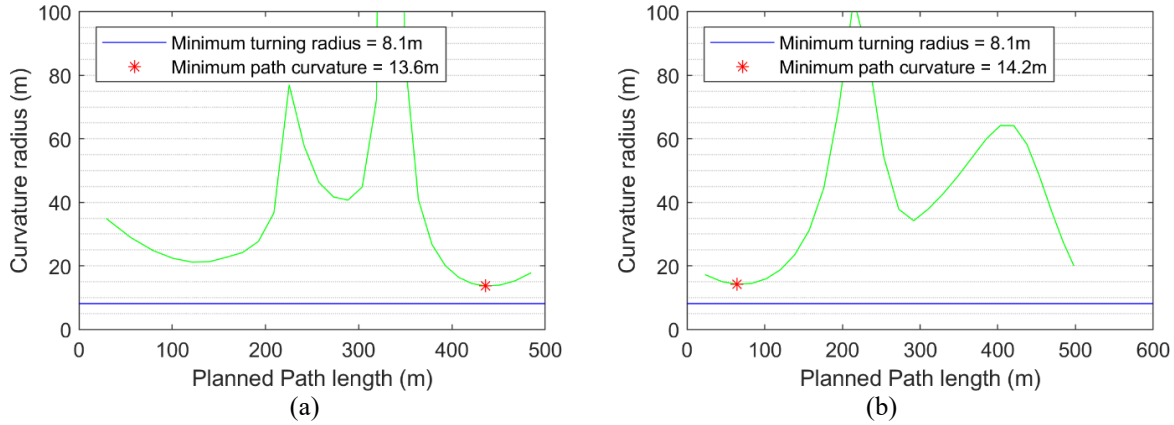


Figure 4: Curvature radius of planned paths for (a) 2D and (b) 3D

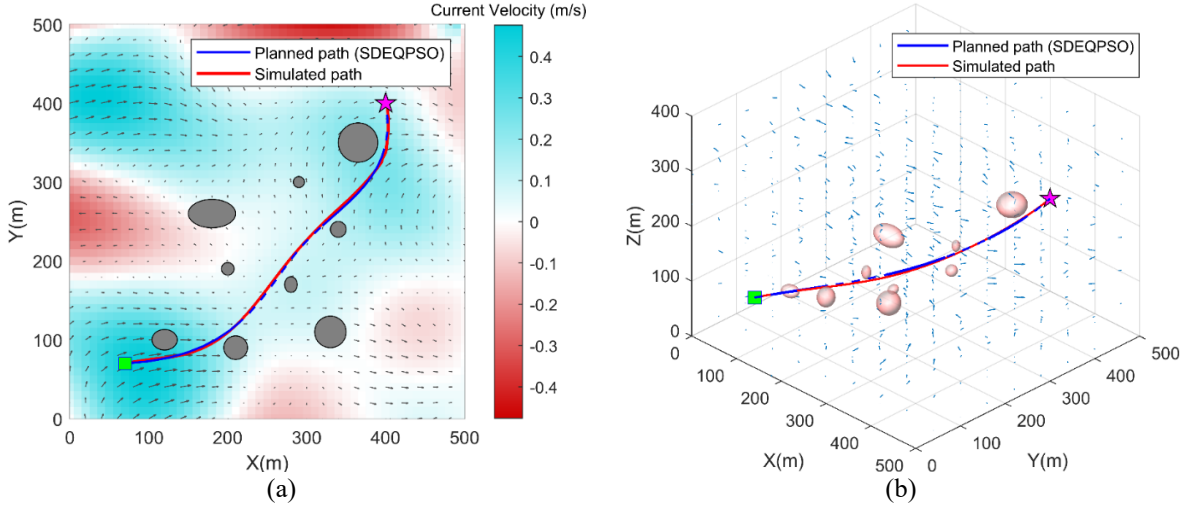


Figure 5: Validation of path solution in (a) 2D scenario and (b) 3D scenario

The cross-track errors of the simulated paths relative to the planned paths for the 2D and 3D scenarios are graphed in Figure 6. The errors for both scenarios were well below 1 metre, proving that the AUV was able to follow the planned paths closely. Hence, the simulation results showed that the path solutions generated by the proposed algorithm were smooth and feasible for the path planning application.

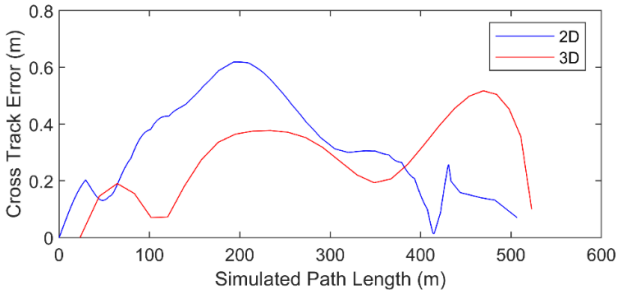


Figure 6: Cross-track error of simulated paths relative to planned paths

7. CONCLUSION

By selectively hybridizing with differential evolution, this paper presents new variants of PSO with improved search ability for the global minimum path of an AUV without increasing the computational requirements. The proposed algorithms were benchmarked against other algorithms in an offline AUV path planner because if the proposed algorithms can provide better computational efficiency to demonstrate the

minimum capability of a path planner, then they can outperform the tested algorithms in the online path planner. Based on the Monte Carlo simulations and ANOVA procedures, the SDEAPSO and SDEQPSO algorithms were able to achieve similar performance to DEPSO and DEQPSO algorithms in terms of solution quality and stability, while having a significantly lower computational requirement. Most importantly, the simulation results showed that the planned paths in both the 2D and 3D scenarios were smooth, feasible and able to account for a priori known environment.

The PSO-based algorithms proposed in this study are most efficient for solving non-deterministic polynomial-time (NP)-hard problem, such as the path planning problem. Although the simulation assumed a priori known environment to represent the minimum capability of a path planner, the algorithms can be adapted to a more realistic operational condition in future work due to the demonstrably high computational efficiency, which is suitable for solving compute-intensive problems such as path re-planning in highly dynamic environments. The future extension of this work can include developing a path re-planning algorithm that can deal with a priori unknown environment.

ACKNOWLEDGEMENT

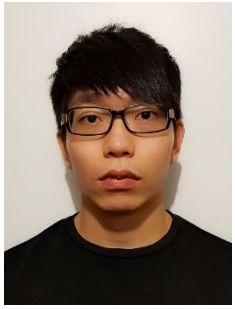
The present work is supported by the Tasmania Graduate Research Scholarship provided by Australia Maritime College (AMC). The authors acknowledge the AUV team in AMC for providing the data from their field experiment at Beauty Point, Tasmania, Australia.

REFERENCES

[1] T. Xue *et al.*, "Trajectory planning for autonomous mobile robot using a hybrid improved QPSO algorithm," *Soft Computing,* vol. 21, no. 9, pp. 2421-2437, 2017.
[2] Z. Zeng *et al.*, "A comparison of optimization techniques for AUV path planning in environments with ocean currents," *Robotics and Autonomous Systems,* vol. 82, pp. 61–72, 2016.
[3] D. Youakim and P. Ridao, "Motion planning survey for autonomous mobile manipulators underwater manipulator case study," *Robotics and Autonomous Systems,* vol. 107, pp. 20–44, 2018.
[4] D. Kruger *et al.*, "Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 2007: IEEE, pp. 4265–4270.
[5] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The Field D* algorithm," *Journal of Field Robotics,* vol. 23, no. 2, pp. 79-101, 2006.
[6] C. Petres *et al.*, "Path Planning for Autonomous Underwater Vehicles," *IEEE Transactions on Robotics,* vol. 23, no. 2, pp. 331-341, 2007.
[7] D. Rao and S. B. Williams, "Large-scale path planning for underwater gliders in ocean currents," in *Australasian Conference on Robotics and Automation (ACRA)*, Sydney, Australia, 2009, pp. 2–4.
[8] J. D. Hernández *et al.*, "Online motion planning for unexplored underwater environments using autonomous underwater vehicles," *Journal of Field Robotics,* vol. 36, no. 2, pp. 370–396, 2019.
[9] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE Journal of Oceanic Engineering,* vol. 29, no. 2, pp. 418–429, 2004.
[10] J. Witt and M. Dunbabin, "Go with the flow: Optimal AUV path planning in coastal environments," in *Australasian Conference on Robotics and Automation*, Canberra, Australia, 2008, no. 2, pp. 1–9.
[11] J. Sun, C. H. Lai, and X. J. Wu, *Particle swarm optimisation classical and quantum perspectives*. Boca Raton, FL: CRC Press (in English), 2012.
[12] Y. Qin *et al.*, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *International Conference on Machine Learning and Cybernetics*, Shanghai, China, 2004, vol. 4, pp. 2473-2478.
[13] Y. Fu *et al.*, "Path planning for UAV based on quantum-behaved particle swarm optimization," in *Proceedings of SPIE - The International Society for Optical Engineering*, 2009,
[14] Z. B. Shi *et al.*, "Path planning for mobile robot based on quantum-behaved particle swarm optimization," *Journal of Harbin Institute of Technology,* Article vol. 42, no. 2, pp. 33-37, 2010.
[15] Z. Zeng *et al.*, "Efficient path re-planning for AUVs operating in spatiotemporal currents," *Journal of Intelligent & Robotic Systems,* vol. 79, no. 1, pp. 135–153, 2015.
[16] S. MahmoudZadeh *et al.*, "Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms," *Applied Soft Computing,* vol. 70, pp. 929–945, 2018.
[17] H. S. Lim *et al.*, "Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner," in *IEEE OES Autonomous Underwater Vehicle Symposium*, Porto, Portugal, 2018: IEEE, pp. 1–7.
[18] Y. Fu *et al.*, "Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 43, no. 6, pp. 1451–1465, 2013.
[19] Z. H. Zhan *et al.*, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 39, no. 6, pp. 1362-1381, 2009.
[20] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *6th International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995: IEEE, pp. 39-43.

[21] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Congress on Evolutionary Computation*, Washington, DC, 1999, vol. 3: IEEE, pp. 1945-1950.
[22] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *IEEE International Conference on Evolutionary Computation*, Anchorage, AK, 1998: IEEE, pp. 69-73.
[23] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *International Conference on Evolutionary Programming*, San Diego, CA, 1998: Springer, pp. 591-600.
[24] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Congress on Evolutionary Computation*, La Jolla, CA, 2000, vol. 1: IEEE, pp. 84-88.
[25] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Congress on Evolutionary Computation*, Seoul, South Korea, 2001, vol. 1: IEEE, pp. 81-86.
[26] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Congress on Evolutionary Computation*, 2004, vol. 1: IEEE, pp. 325-331.
[27] W. J. Zhang and X. F. Xie, "DEPSO: hybrid particle swarm with differential evolution operator," in *IEEE International Conference on Systems, Man and Cybernetics*, 2003, vol. 4: IEEE, pp. 3816-3821.
[28] B. Raphael and I. F. Smith, *Fundamentals of computer-aided engineering*. Chichester, United Kingdom: John Wiley & Sons, 2003.
[29] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation,* vol. 1, no. 1, pp. 67-82, 1997.
[30] M. Locatelli, "A note on the Griewank test function," *Journal of global optimization,* vol. 25, no. 2, pp. 169-174, 2003.
[31] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel computing,* vol. 17, no. 6-7, pp. 619-632, 1991.
[32] D. H. Ackley, "The model," in *A Connectionist Machine for Genetic Hillclimbing*: Springer, 1987, pp. 29-70.
[33] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation,* vol. 1, no. 1, pp. 1-23, 1993.
[34] P. E. McKight and J. Najab, "Kruskal-Wallis test," *The Corsini encyclopedia of psychology,* 2010.
[35] Y. Hochberg and A. C. Tamhane, *Multiple Comparison Procedures*. New York, NY: John Wiley & Sons, 1987.
[36] L. Piegl and W. Tiller, *The NURBS book*. Berlin: Springer Science & Business Media, 2012.
[37] C. De Boor *et al.*, *A practical guide to splines*. Springer-Verlag New York, 1978.
[38] S. Mirjalili, J. Song Dong, and A. Lewis. *Ant colony optimizer: Theory, literature review, and application in AUV path planning, Studies in Computational Intelligence*, vol. 811, pp. 7-21, 2020.
[39] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. Norway: John Wiley & Sons, 1999.
[40] T. T. J. Prestero, "Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle," Massachusetts institute of technology, 2001.
[41] W. Caharija *et al.*, "Path following of underactuated autonomous underwater vehicles in the presence of ocean currents," in *IEEE Conference on Decision and Control*, Maui, HI, 2012: IEEE, pp. 528–535.
[42] Y. Eng *et al.*, "Online system identification of an autonomous underwater vehicle via in-field experiments," *IEEE Journal of Oceanic Engineering,* vol. 41, no. 1, pp. 5-17, 2016.

BIOGRAPHIES OF AUTHORS



Hui Sheng Lim received a Bachelor's degree in Marine and Offshore Engineering in 2016 from the University of Tasmania, Australia, where he is currently working toward a PhD degree. His current research interests include optimal guidance, navigation and control system of autonomous underwater vehicles (AUV) using swarm intelligence.

Shuangshuang Fan received a B.E. in Mechanical Engineering from Shandong University, Jinan, China, in 2008, and a PhD in Mechatronic Engineering from Zhejiang University, Hangzhou, China, in 2013. From 2013 to 2014, she was a Research Engineer with the Institute of Shanghai Aerospace Control Technology. She was with the Acoustic Signal Processing Lab, Zhejiang University, as a Postdoctoral Researcher from 2014 to 2017. She was a lecturer at Australian Maritime College, University of Tasmania. Her research interests include the navigation, control and path planning of underwater vehicles in dynamic environments.

Christopher Chin commenced his academic career as a lecturer at RMIT University whilst undertaking his doctoral studies in Mathematics in 2003. He is currently a mathematician and a senior lecturer at the University of Tasmania, Australia. He works within the National Centre of Maritime Engineering and Hydrodynamics at the Australian Maritime College. He works closely with the Maritime Engineers focusing on Maritime Education and alternative energies. He is currently focusing on investigating the potential use of alternative energies in the maritime offshore sector in order to address the depletion of fossil fuels.

Shuhong Chai completed her Masters Degree in Naval Architecture at the Dalian University of Technology in China. In 2006 she obtained her doctoral degree from Universities of Strathclyde and Glasgow. She was then a senior hydrodynamics consultant and project manager with Oceanic Consulting Corporation in Canada. In 2008, Dr Chai joined AMC as a Senior Lecturer. She has since undertaken senior administrative roles including Director of the National Centre for Maritime Engineering and Hydrodynamics, and Associate Dean-Learning and Teaching for the AMC, Associate Dean Global of College of Sciences and Engineering and AMC Principal. She is very active in the subsea and underwater technology field. She is recently appointed as a member of the Committee of Loads of International Ship and Offshore Structures Congress (ISSC) from 2018 to 2021. She has been a specialist member of Committee of Subsea Technology of ISSC from 2015 to 2018 and Committee of Risers and Pipelines of ISSC from 2012 to 2015.

Neil Bose is the Vice President (Research) at Memorial University, Newfoundland and Labrador's University. Previously he was Principal of the Australian Maritime College (AMC), a specialist institute of the University of Tasmania, and a Professor of Maritime Hydrodynamics. Neil obtained his B.Sc. in Naval Architecture and Ocean Engineering from the University of Glasgow in 1978 and his PhD also from Glasgow in 1982. He came to AMC in Tasmania in May 2007 as the Manager of the Australian Maritime Hydrodynamics Research Centre. His personal research interests are in marine propulsion, autonomous underwater vehicles, ocean environmental monitoring, ocean renewable energy, ice/propeller interaction and aspects of offshore design. Neil Bose is an ocean engineer and naval architect with an international reputation in marine propulsion built up through close collaboration with international industry.

Computer Modeling in Engineering & Sciences
DOI:10.32604/cmes.2020.011648
Article

Online AUV Path Replanning Using Quantum-Behaved Particle Swarm Optimization with Selective Differential Evolution

Hui Sheng Lim[1,*], Christopher K. H. Chin[1], Shuhong Chai[1] and Neil Bose[1,2]

[1]National Centre for Maritime Engineering and Hydrodynamics, Australian Maritime College, University of Tasmania, Launceston, TAS, 7250, Australia
[2]Memorial University of Newfoundland, St. John's, NL, A1C 5S7, Canada
[*]Corresponding Author: Hui Sheng Lim. Email: hui.lim@utas.edu.au
Received: 29 May 2020; Accepted: 27 July 2020

Abstract: This paper presents an online AUV (autonomous underwater vehicle) path planner that employs path replanning approach and the SDEQPSO (selective differential evolution-hybridized quantum-behaved particle swarm optimization) algorithm to optimize an AUV mission conducted in an unknown, dynamic and cluttered ocean environment. The proposed path replanner considered the effect of ocean currents in path optimization to generate a Pareto-optimal path that guides the AUV to its target within minimum time. The optimization was based on the onboard sensor data measured from the environment, which consists of a priori unknown dynamic obstacles and spatiotemporal currents. Different sensor arrangements for the forward-looking sonar and horizontal acoustic Doppler current profiler (H-ADCP) were considered in 2D and 3D simulations. Based on the simulation results, the SDEQPSO path replanner was found to be capable of generating a time-optimal path that offered up to 13% reduction in travel time compared to the situation where the vehicle simply followed a path with the shortest distance. The proposed replanning technique also showed consistently better performance over a reactive path planner in terms of solution quality, stability, and computational efficiency. Robustness of the replanner was verified under stochastic process using the Monte Carlo method. The generated path fulfilled the vehicle's safety and physical constraints, while intelligently exploiting ocean currents to improve the vehicle's efficiency.

Keywords: Autonomous underwater vehicle; path planning; particle swarm optimization; sonar detection; Monte Carlo methods

1 Introduction

AUVs have become an increasingly important tool for performing various operations, ranging from seabed surveys, coastal mapping, and environmental monitoring for scientific research purposes, to anti-submarine warfare for defense purposes. To date, most of the research has been dedicated to improving the autonomy of the AUVs in order to enable operation with longer endurance, in which the vehicles may come across unknown obstacles and large-scale time-varying ocean circulation. Strong ocean currents and eddies may push an AUV off its planned path, causing a profound impact on the vehicle's

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

performance, particularly its battery consumption and thus the vehicle's endurance. Therefore, it is important for an AUV path planner to take into consideration the effect of ocean currents [1]. By adapting its planning to ocean currents, a path planner can enable an AUV to surf the favorable currents that assist the vehicle's motion, while avoiding the adverse currents that are opposing it. This paper proposes a novel path replanner that generates time-optimal paths to exploit ocean currents in a fully unexplored and dynamic environment. The path replanner used an efficient SDEQPSO algorithm to achieve a balance between the generated path quality and its computational load [2], which is a crucial factor for succeeding in long-endurance missions.

Planning an AUV path in an unknown, dynamic, and cluttered underwater environment is a multi-objective and multimodal optimization problem, which requires a computationally efficient algorithm. Recent comparison studies [2–4] discussed various path planning techniques including Artificial Potential Field (APF), graphical search methods, sampling-based methods, and metaheuristic optimization. APF [5] is efficient for high dimensional problems, but it is highly vulnerable to local minima. Search-based methods such as A* [1,6] and Field D* Lite [7] are low complexity algorithms with applications limited to lower-dimensional and less complex problems. Rapidly exploring random tree (RRT) [8] and its variant RRT* [9] are sampling-based methods that can be applied for high dimensional and time-constrained scenarios, but the generated paths are usually sub-optimal and require further refinement. Metaheuristic optimization techniques such as genetic algorithm [10] and particle swarm optimization (PSO) [11] are efficient for complex multimodal path planning problems and have higher resistance towards local minima. Among the existing metaheuristic algorithms, the quantum-behaved PSO (QPSO) algorithm and its variants were found to have outstanding performance in terms of robustness and solution quality for solving the AUV path planning problem [3,12].

QPSO-based path planner is suitable for dynamic environments where real-time/online planning of the trajectory is required because it can maintain a large pool of solutions, which is available at any time during the mission. These solutions can serve as the initial solutions whenever the replanning of a path is needed, thus significantly improving the computational efficiency. Nevertheless, the algorithm may converge at local minimum solutions if the time allowed for path planning is limited, which is often the case in real AUV operations. Some have proposed methods to improve their resistance to local minima but at the cost of computational load [13]. Based on a recent comparison study [12] on the variants of the QPSO algorithm, selectively Differential Evolution (DE)-hybridized QPSO (SDEQPSO) was developed through the hybridization of DE operation in the QPSO algorithm using a selective scheme [2]. By benchmarking against other metaheuristic path planners including standard DE, PSO, and other DE-hybridized algorithms, the SDEQPSO algorithm was found to have improved search ability for the global optimal path and provide higher resistance to local minima with an insignificant increase in its computational requirement. It demonstrated the ability to generate high-quality AUV paths without imposing a high computational load on the vehicle's computer.

There are various existing techniques used to perform online path planning in an unexplored and dynamic ocean environment. The traditional approach is known as reactive path planning, which generates a new path reactively to adapt to the varying environment, while the previously planned path is discarded. To achieve online path planning while accounting for the effect of ocean currents, the reactive approach can be combined with various algorithms such as genetic algorithm (GA) [10], APF [5], level set methods [14], and swarm optimization [15]. A different approach of online path planning [16] combined path following and obstacles avoidance control to handle dynamic environment efficiently but at the cost of path quality. A similar study [17] was able to improve the path quality by combining fuzzy control and QPSO. However, this approach does not incorporate the effect of ocean currents in the path planner.

In contrast to reactive path planning, an approach known as the path replanning scheme generates a new path based on the previous solution [18]. It is deemed more computational efficient if the optimized path can

be generated by modifying the previously planned path because there is a high possibility that the new optimized path nodes can be placed near to the previous solution. This is because the ocean environmental conditions usually vary gradually over time, and therefore the new environmental conditions may resemble the conditions in the previous planning cycle to some extent. The increase in computational efficiency by making use of the previous solutions can be significant especially when the search space is vast and highly dynamic. Some existing path replanners [7,9,19] allow replanning based on a single previous solution. Path replanning can be achieved more efficiently by using a population-based optimization algorithm such as the QPSO, which can maintain all previous solutions at every iteration. For example, a QPSO-based path replanner [18] was proposed to replan the path of an AUV in a spatiotemporal environment at a predefined fixed interval. This path replanner has a high requirement for onboard sensor configuration because it requires either the global environmental information or all the information surrounding the AUV up to a certain radius to be obtained in real-time.

In this study, the novel SDEQPSO path replanner generates a time-optimal path for an AUV by adapting its solutions to ocean currents and intelligently using the currents to improve the vehicle's efficiency. Based on the onboard sensor measurements, the path replanner continuously generates the AUV path at an adaptive interval to react against the environmental changes. Different sensor configurations were considered and compared while assuming the measurements to be noise-free and reliable. The mission scenario with a priori unknown dynamic obstacles and spatiotemporal currents was first simulated in a 2D domain, followed by the simulation in a 3D domain. Monte Carlo method was used to establish a comprehensive evaluation study for the proposed path replanner. The proposed path replanner offers the following advantages:

1. It generates time-optimal paths by using a computationally efficient algorithm to improve an AUV's performance.
2. It accounts for obstacle avoidance, the spatiotemporal variability of ocean environments, and the constraints imposed by missions and vehicles.
3. No pre-generated path is required.
4. It demonstrates its scalability for missions that require different setups of onboard sensors.

This paper is organized as follows. Section 2 describes the formulation of the path planning problem. An overview of the SDEQPSO algorithm is provided in Section 3. The AUV simulation model used in this paper is outlined in Section 4. In Section 5, the simulation setup, results, and discussions are presented. Finally, the paper is concluded in Section 6 along with future research directions.

2 Problem Formulation for Path Replanning

2.1 Path Formulation

In this paper, the primary objective of the AUV path planner was to solve a multi-objective multimodal optimization problem, which was required to determine an optimal path that can guide the AUV towards its target through an ocean environment. A feasible path of the AUV can comprise a series of nodes along the path from the starting point to the endpoint (target). An optimal path can be obtained by controlling and optimizing the node coordinates. The optimization does not involve the starting point and the endpoint of the path because the same start and end locations are shared by all potential paths.

In the SDEQPSO path planner, each particle in the swarm represents a potential path solution. A swarm population containing N particles can be written as a matrix $X = [X_1, X_2,\ldots, X_N]^T$, where X denotes the position vector of the particles. Entries of the particles' position vectors represent the node coordinates. Given that every path comprises $n + 2$ nodes (including the starting point and endpoint), the path optimization involves n number of nodes. The position vector of a 2D particle requires $2n$ dimensions to register the polar coordinates of n node(s); this includes n dimension(s) for radial coordinates r and n

dimension(s) for azimuthal angle coordinates φ. Meanwhile, a 3D particle requires $3n$ dimensions to record n node(s) in the spherical coordinates, which include extra n dimension(s) for polar angle coordinates θ. The ith particle's position vectors at tth iteration for 2D and 3D can be written as Eqs. (1) and (2) respectively.

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,n}^t, x_{i,n+1}^t, \ldots, x_{i,2n}^t\right], \qquad i \in \{1, 2, \ldots, N\} \tag{1}$$

$$X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,n}^t, x_{i,n+1}^t, \ldots, x_{i,3n}^t\right], \qquad i \in \{1, 2, \ldots, N\} \tag{2}$$

Cartesian coordinates of the first path node can be obtained from the polar coordinates (2D) by using Eq. (3) and from spherical coordinates (3D) by using Eq. (4).

$$\begin{aligned} x_{i,1} &= r_{i,1} \cos \varphi_{i,n+1} \\ y_{i,1} &= r_{i,1} \sin \varphi_{i,n+1} \end{aligned} \tag{3}$$

$$\begin{aligned} x_{i,1} &= r_{i,1} \cos \varphi_{i,n+1} \sin \theta_{i,2n+1} \\ y_{i,1} &= r_{i,1} \sin \varphi_{i,n+1} \sin \theta_{i,2n+1} \\ z_{i,1} &= r_{i,1} \cos \theta_{i,2n+1} \end{aligned} \tag{4}$$

B-spline geometry was applied to generate an AUV path from the path nodes. B-spline is a parametric curve generated from a series of connected piecewise polynomials [20], which are suitable for modeling the AUV path because of its continuity for a smooth path and locality for altering the path without affecting continuity. Based on the curve function in Eq. (5), B-spline curve can be constructed by using the path nodes as the control points to produce an output vector $P(u)$, which represents a $k + 1$ order B-spline curve in the form of discretized waypoints. Assuming $n + 2$ number of control points is involved, the number of piecewise polynomials that can be generated is $n + 1$.

$$P(u) = \sum_{i=0}^{n+1} x_i B_{i,k}(u), \qquad i \in \{0, 1, 2, \ldots, n+1\} \tag{5}$$

where x_i denotes the control points and u is a strictly increasing knot sequence from a knot vector $U = [u_0, \ldots, u_i, \ldots, u_{n+k+2}]$. $B_{i,k}(u)$ is the basis functions for piecewise polynomial of k degree, which can be obtained from Cox de Boor recursion [20] as follows.

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \tag{7}$$

The continuity of the spline is fully dependent on the basis functions. Hence, the control points, i.e., path nodes can be adjusted during the path optimization process without affecting the spline continuity.

2.2 Forward-Looking Sonar Model

A forward-looking sonar (FLS) model was used in the simulation for the detection of obstacles. The settings of the FLS model was specified as follows: 80 m detection range, 120° field of view, 121 number of beams (with 1° separation between beams), and 100 Hz detection frequency.

Generally, the sonar configuration of an AUV can vary depending on the mission requirements. The horizontal sonar configuration, in which the fan-shaped FLS model was installed in such a way that the

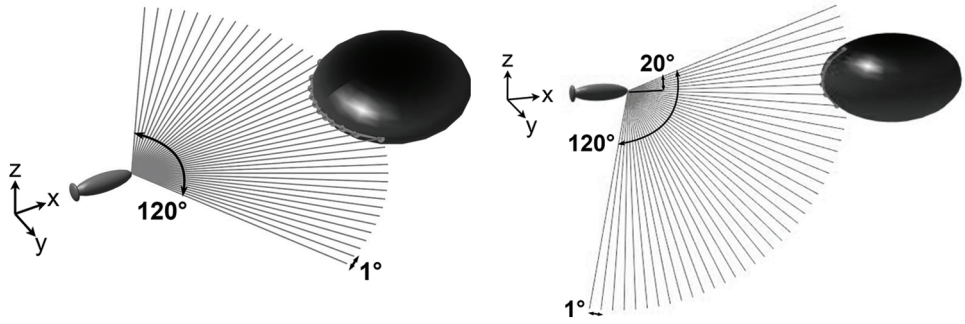


Figure 1: Forward-looking sonar configured in horizontal plane (left) and vertical plane (right)

sonar fan aligns with the horizontal plane of the vehicle, is suitable for missions such as area coverage survey. Some missions such as operations underneath ice shelves or near-seabed operations require the FLS to be configured in the vertical plane. Therefore, two sonar configurations were considered in this study as shown in Fig. 1. The vertical sonar configuration has an offset of 20° above the horizontal plane.

All obstacles in the simulated space were configured to be irregular and a priori unknown. The coordinates indicating the boundaries of the obstacles were generated by sonar detection. This information was then sent to the path planner for path computation.

2.3 Current Profiler Model

In order to allow the adaptation of path solutions to ocean currents, real-time current information based on the simulated measurement from a forward-looking horizontal acoustic Doppler current profiler (H-ADCP) was fed to the path planner. The path planning simulation used a 300 kHz H-ADCP with 200 m detection range, which is able to reconstruct a velocity profile of 200 m × 50 m in the looking-forward region of the vehicle [21].

2.4 Objective Functions

The implementation of PSO-based algorithms in an optimization problem requires the development of objective functions to evaluate the particles' fitness based on their respective solutions. Objective functions usually account for most of the computational time as PSO-based algorithms are computationally efficient [22]. Objective functions should be developed in accordance with the optimization criteria of the problem. In order to produce an accurate fitness representation model for finding the optimal solution, the developed functions must closely resemble the physical conditions of the problem space. Path planning is a minimization problem that requires the AUV travel time to be minimized. Therefore, its optimal solution should have the lowest cost/fitness. The multi-objective path planning problem was modeled using a single aggregate objective function with equal weights assigned to the underlying objective functions F_k. Thus, the optimal solution $X^{\dagger}$ for the path planning problem can be given by the function in Eq. (8).

$$X^{\dagger} = \arg\min \sum_{k=1}^{2} F_k(X_i) \tag{8}$$

The first objective function F_1 was developed to measure the particles' fitness with respect to the time required to travel on the corresponding paths while considering the effect of ocean currents. This allowed the path planner to determine the time-optimal path, which would guide the vehicle to its destination within minimum time. After constructing a B-spline path from the path nodes, the path X_i can be represented in the form of discretized waypoints $P = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$, where P is generated from the B-spline function, and m is the total number of discretized waypoints. For the ith particle, its travel time cost

$F_1(X_i)$ can be given as the sum of discretized time required to travel on each small path segment that links the consecutive discretized waypoints in P as shown in Eq. (9).

$$F_1(X_i) = \sum_{j=1}^{m-1} \frac{\left\| \overrightarrow{p_{i,j}p_{i,j+1}} \right\|}{|V_g|}, \qquad j \in \{1, 2, \ldots, m-1\} \tag{9}$$

where V_g is the AUV's resultant ground reference velocity, which is the resultant AUV velocity under the influence of surrounding ocean currents. Projection of the current velocity V_c onto the vector of AUV water reference velocity V_a, which is in the same direction of the path vector, allowed the effect of currents on the AUV to be determined. Thus, V_g can be given by the sum of V_a and the contribution of V_c as shown in Eq. (10). Eq. (10) enabled the path planner to adapt its solutions to the measured currents.

$$V_g = V_a + \frac{V_c \cdot \overrightarrow{p_{i,j}p_{i,j+1}}}{\left\| \overrightarrow{p_{i,j}p_{i,j+1}} \right\|} \tag{10}$$

In order to generate a collision-free and feasible path, solutions generated by the algorithm were required to satisfy the following objectives and boundaries:

- *Obstacle avoidance:* Maintain a safe distance with obstacles to prevent collisions.
- *Radial boundary:* Control the placement of path nodes for path replanning.
- *Azimuthal boundary:* Constrain the solutions with respect to the AUV's minimum turning radius.
- *Polar boundary:* Constrain the solutions with respect to the AUV's pitch control limitation.

The path solutions were constrained based on a setup discussed in the previous work [23], which are explained as follows. The second objective function F_2 was designed as a penalty function for achieving obstacle avoidance. The penalty function measured the threat cost of a given path with respect to its exposure to threats/obstacles. Threat detection points, which were generated by the forward-looking sonar, were treated as circles under the 2D condition and as spheres under 3D. The radii of the threat circles/spheres were set as the safety clearance required by the AUV to maintain a safe distance with the threats. The threat cost can be obtained by checking the path's intersection with the threat circles/spheres. A threat h in 3D with a detection point $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and safety clearance $O_{r,h}$ can be represented by a parametric equation in Eq. (11). A path segment that connects two adjacent waypoints $p_{i,\,j} = (x_1, y_1, z_1)$ and $p_{i,\,j+1} = (x_2, y_2, z_2)$ can also be expressed as shown in Eq. (12).

$$(x - O_{cx})^2 + (y - O_{cy})^2 + (z - O_{cz})^2 = {O_r}^2 \tag{11}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + s \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{12}$$

Substituting Eq. (12) into Eq. (11) produced the following equations, which are expressed in terms of s. The intersection between the path and the threat can be checked by computing the discriminant ξ of Eq. (13) by using Eq. (17).

$$As^2 + Bs + C = 0 \tag{13}$$

$$A = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \tag{14}$$

$$B = 2\left[(O_{cx} - x_1)(x_1 - x_2) + (O_{cy} - y_1)(y_1 - y_2) + (O_{cz} - z_1)(z_1 - z_2)\right] \tag{15}$$

$$C = (O_{cx} - x)^2 + (O_{cy} - y)^2 + (O_{cz} - z)^2 - {O_r}^2 \tag{16}$$

$$\zeta = B^2 - 4AC \tag{17}$$

There will be no intersection between the path and the threat when $\xi = 0$, i.e., the path is tangent to the threat region. When $\xi > 0$, collisions between the AUV path and the threat will occur if the roots s_1 and s_2 given by Eq. (18) are within the range of $0 \le s_1, s_2 \le 1$.

$$s_1, s_2 = \frac{-B \pm \sqrt{\xi}}{2A} \tag{18}$$

If there is a collision, the threat cost $F_2(X_i)$ can be obtained from Eq. (19), which was developed to be directly proportional to the length of the path segment contained in the threat region. The intersection points S_1 and S_2 can be determined by solving Eq. (13) using the obtained s_1 and s_2 and substituting them back into Eq. (12).

$$F_2(X_i) = \sum_{h=1}^{H} \sum_{j=1}^{m-1} \frac{\left\| \overrightarrow{S_1 S_2} \right\|}{2O_{r,h}} \tag{19}$$

In order to improve the computational efficiency during path replanning, the placement of the path nodes was constrained by the radial boundary. Each path node was constrained to be placed within a concentric annulus, which is the region bounded by a pair of adjacent concentric circles with different radii. The radii were defined by a lower boundary $R_{\min}$ and an upper boundary $R_{\max}$ as defined in Eq. (20). The search domains of radial coordinates were hard-constrained between the two boundaries.

$$\begin{aligned} R_{\min} &= \left[0, r_d, 2r_d, \ldots, r_{target}\right] \\ R_{\max} &= \left[r_d, 2r_d, 3r_d, \ldots, r_{target}\right] \end{aligned} \tag{20}$$

where r_d is defined as the radial distance between two concentric circles and r_{target} denotes the radial coordinate of the target. Path nodes exceeding the boundaries $R_{\min}$ and $R_{\max}$ will be regenerated. The total number of path nodes n required for generating the path can be controlled by r_d as defined by Eq. (21).

$$n = \left\lceil r_{target} / r_d \right\rceil. \tag{21}$$

Hard constraints were also applied to ensure the generated path solutions respect the minimum turning radius and the pitch limitation of the AUV. An azimuthal boundary φ_{max} and a polar boundary θ_{max} were used to constrain the search domain of azimuthal angle coordinate and polar angle coordinate. The path solution will fulfill the constraints if $|\varphi_{i,j}| < \varphi_{max}$ and $|\theta_{i,j}| < \theta_{max}$; otherwise, the solution will be regenerated.

2.5 Path Replanning Scheme

A path replanning scheme was employed in this paper to handle a real-time path planning scenario in a fully unknown and dynamic ocean environment. The SDEQPSO algorithm can maintain the entire population of previous solutions that can be used for replanning the path at any time throughout the mission. The path replanning process was carried out online and continuously at an adaptive interval while the AUV navigates towards its target. The adaptive replanning interval was designed to be reactive to the ocean environment, meaning that a previously optimized path will be replanned when it is unsafe or less optimal due to environmental changes. Flags that will trigger path replanning are:

- Elapsed time since the previous plan exceeds a preset threshold.
- Unexpected obstacles are detected within the safe zone of the vehicle.
- Detected obstacles intersect the previously planned path.

During path replanning, the SDEQPSO path replanner modified the previous solutions to generate a new path that is optimized for a continuously varying environment. The process of reusing the previous solution

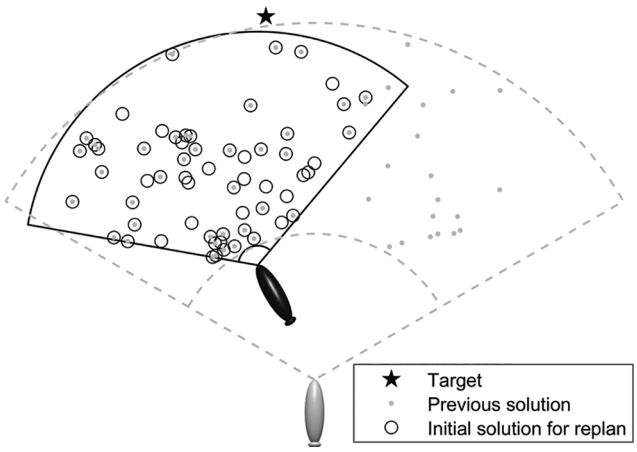


Figure 2: Reuse of solutions in path replanning process (grey vehicle denotes the previous position and black vehicle denotes the current position)

for path replanning can be described by Fig. 2, in which the grey dots and black circles represent the population of waypoints that can be used to construct the AUV path. A portion of the previous solution (grey dots) can be retained and used as the initial population (black circles) for replanning the path. The initialization of path replanning began with defining the new boundary conditions, including $R_{\min}$, $R_{\max}$, φ_{max}, and θ_{max}, based on the new starting point, which is the AUV's current position. The path waypoints behind the new starting point were removed. Next, solutions that satisfy the new boundary conditions were retained, while solutions that violate the boundary conditions were regenerated. The initialized solutions then underwent the SDEQPSO iteration to determine the optimized path.

3 SDEQPSO Algorithm

The SDEQPSO algorithm is based on the QPSO algorithm, which consists of quantum-behaved particles that search for feasible solutions by moving within a multidimensional search space. The solutions are recorded as the particles' positions. For an algorithm that contains N particles with D dimensions for solving an objective function f, the ith particle at tth iteration has the following position vector:

$$X_i^t = \left[x_{i1}^t, x_{i2}^t, \ldots, x_{ij}^t, \ldots, x_{iD}^t\right], \qquad i \in \{1, 2, \ldots, N\} \tag{22}$$

The quantum particles are assumed to be attracted to a 1-dimensional delta potential well centered at a local attraction point for each dimension of the particles' positions. In the quantum state, the momentum and energy of the particles are characterized by a wave function, and thus the position and velocity update equations of the QPSO algorithm are different from the traditional update equations in PSO. Based on the statistical interpretation of the wave function, the probability distribution function of the particles' positions can be obtained to transform the particles' positions from quantum state to classical state by employing Monte Carlo inverse transformation [22]. Accordingly, the position of the ith particle can be updated using the following stochastic equation:

$$X_i^{t+1} = \begin{cases} p_i^t + 0.5 \cdot L_i^t \cdot \ln\left(1/u_i^t\right), & \text{if } u \geq 0.5 \\ p_i^t - 0.5 \cdot L_i^t \cdot \ln\left(1/u_i^t\right), & \text{if } u < 0.5 \end{cases} \tag{23}$$

where u is a uniform distributed random positive number that is less than 1.0, p is the local attractor as defined in Eq. (24), and L is the delta potential well characteristic length as defined in Eq. (25). p and L are based on the particles' previous best position *pbest*, mean best position *mbest*, and the global best position in the swarm *gbest* as defined by Eqs. (26)–(28) respectively.

$$p_i^t = \varphi_i^t \cdot pbest_i^t + \left(1 - \varphi_i^t\right) \cdot gbest^t \tag{24}$$

$$L_i^t = 2 \cdot \beta \cdot \left| mbest^t - X_i^t \right| \tag{25}$$

$$pbest_i^t = \begin{cases} pbest_i^{t-1}, & iff(X_i^t) \geq f(pbest_i^{t-1}) \\ X_i^t, & iff(X_i^t) < f(pbest_i^{t-1}) \end{cases} \tag{26}$$

$$mbest^t = \sum_{i=1}^{N} pbest_i^t / N \tag{27}$$

$$gbest^t = \arg\min\left[f(pbest_i^t)\right] \tag{28}$$

The coefficient φ in Eq. (24) is a uniform distributed random positive number that is less than 1.0. The parameter β in Eq. (25) is known as the contraction-expansion (CE) coefficient. Combining Eqs. (23)–(25) yields the following position update equation for the particles.

$$X_i^{t+1} = \varphi_i^t \cdot pbest_i^t + \left(1 - \varphi_i^t\right) \cdot gbest^t \pm \beta \cdot \left| mbest^t - X_i^t \right| \cdot \ln\left(1/u_i^t\right) \tag{29}$$

Selection of the CE coefficient β is critical for tuning the convergence behavior of the algorithm. As suggested by an empirical study of parameter selection [22], a linearly decreasing β from a maximum value $\beta_{\max}$ of 1.0 to a minimum value $\beta_{\min}$ of 0.5 as shown in Eq. (30) is suitable for most optimization problems.

$$\beta = \beta_{\max} - \frac{t}{t_{\max}}(\beta_{\max} - \beta_{\min}) \tag{30}$$

The SDEQPSO applies DE operation on the particles through a selective scheme to increase swarm diversity and search ability without altering the original particle swarm dynamics. Following the position update operation, the particles are sorted based on their personal best position. The DE operation is applied on a selected number of particles, N_S. A selective factor S is used to control N_S according to Eq. (31).

$$N_S = N \times S, \qquad S \in [0, 1] \tag{31}$$

The DE operation includes mutation, crossover, and selection operators as described below. The mutation and crossover operators will be conducted on the N_S best-performing particles to generate the new vectors, which will replace the N_S worst-performing particles during natural selection.

- Mutation: Eq. (32) is applied to generate a mutated solution vector U.

$$U_i^t = gbest^t + \frac{(pbest_{r1}^t - pbest_{r2}^t) + (pbest_{r3}^t - pbest_{r4}^t)}{2} \tag{32}$$

where r_1, r_2, r_3 and r_4 are random particle indices that are mutually different, and different from the index i of the selected particle and the index of the global best particle, i.e., $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i \neq gbest$.

- Crossover: Eq. (33) performs crossover between the mutated vector and the personal best position of the selected particle to generate a new vector T.

$$T_i^t = \left[\tau_{i1}^t, \ldots, \tau_{ij}^t, \ldots, \tau_{iD}^t\right]$$
$$\tau_{ij}^t = \begin{cases} u_{ij}^t, & \text{if } r_j \leq CR || j = r \\ pbest_{ij}^t, & \text{if } r_j > CR || j \neq r \end{cases} \tag{33}$$

where CR is the crossover probability with a suggested value of 0.85, r_j is a uniformly distributed random number in the range [0,1.0], and r is a random positive integer in the range of 1 to the total number of dimensions, D, contained by the particle.

- Natural selection: The worst-performing particle is replaced by the new vector T. All potentially optimal solutions will not be affected because only the worst-performing particles will be replaced.

For the path planning problem of an AUV, the selective factor S has a suggested value of 0.3 to help increasing swarm diversity and to maintain a sufficient number of potentially best particles [2]. The selective DE operation in SDEQPSO promotes global convergence by improving the particles' evolutionary rate and removing the least desirable solutions. The proposed SDEQPSO path replanner can be implemented according to the following pseudocode.

Step 1. **Define** the settings of algorithm and ocean environment.

Step 2. **Initialize** a group of candidate paths by generating random particle positions in Eq. (22). Define *pbest* as the current particle positions.

Step 3. **Check** the path replanning flag.

Step 4. **If** replanning flag == 1

While the termination criteria are not satisfied,

For t = 1, 2, …, t_{max},

Find *mbest* by using Eq. (27).

Obtain particle fitness $f(X_i^t)$ from the objective function.

Find *pbest* and *gbest* by using Eqs. (26) and (28) respectively.

Calculate β as required.

For each particle i = 1, 2, …, N,

Vary particle position by using Eq. (29).

End

Sort particles based on personal best fitness.

For k = 1, 2,…, N_Sth best performing particle,

Mutation: Generate mutated solution U_k^t according to Eq. (32).

Crossover: Generate trial solution T_k^t according to Eq. (33).

Natural selection: Replace kth worst-performing particle with T_k^t.

End

End

Return *gbest* that contains the optimal path upon algorithm termination.

Else

Follow the previous path.

Step 5. **Back** to Step 3 if the mission is not completed.

---

4 AUV Simulation Model

Simulation of a real-time path planning scenario requires the use of an AUV mathematical model. The SDEQPSO path planner generates the AUV path in real-time based on the feedback from the sensors and the AUV dynamic model as illustrated in Fig. 3. The generated paths were used as the reference trajectory during the simulation of a dynamic model of the Hydroid REMUS 100, 1.7-meter-long torpedo-shaped AUV. This section outlines the dynamic model and the path following controller used.

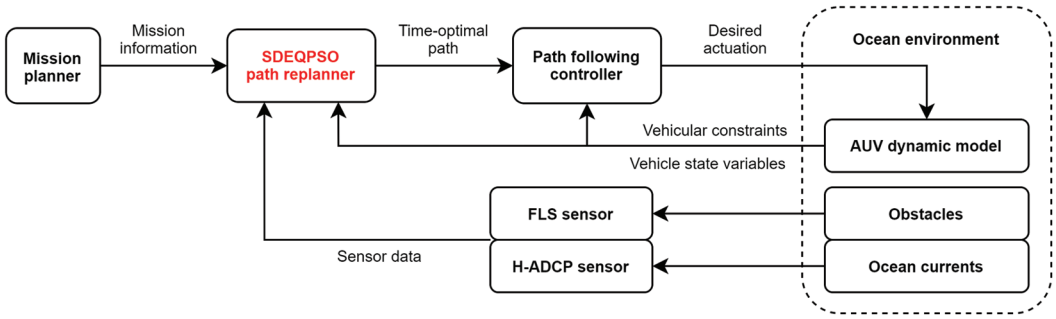


Figure 3: Implementation of SDEQPSO path replanner

4.1 Dynamic Model

The 6 DOF equations of motion of a typical AUV can be derived based on Fossen's vectorial representation [24] and SNAME (Society of Naval Architects and Marine Engineers) formulation as described in Eqs. (34) and (35).

$$\dot{\eta} = \begin{bmatrix} R(\eta_2) & 0_{3\times3} \\ 0_{3\times3} & T(\eta_2) \end{bmatrix} v \tag{34}$$

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \tag{35}$$

where $R(\eta_2)$ denotes the rotation matrix of translational velocities for conversion between inertial and body-fixed reference frames and $T(\eta_2)$ is the rotation matrix of angular velocities. η includes the vehicle's position η_1 and orientation η_2 with respect to the inertial reference frame. The derivative of η in Eq. (34) represents the rate of change of η. v is the matrix that includes the vehicle's translational velocities v_1 and rotational velocities v_2 with respect to the body-fixed reference frame as shown in Eq. (36).

$$\eta = [\eta_1 \quad \eta_2]^{\mathrm{T}} = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^{\mathrm{T}},$$
$$v = [v_1 \quad v_2]^{\mathrm{T}} = [u \quad v \quad w \quad p \quad q \quad r]^{\mathrm{T}}$$

In Eq. (35), M is the inertial matrix of the rigid body and added mass, while $C(v)$ is the Coriolis matrix. $D(v)$ and $g(\eta)$ denote the hydrodynamics damping matrix and the hydrostatics restoring force respectively. The actuators' control forces are included in τ. The mathematical model of the REMUS 100 used in this study was derived from Eqs. (34) to (36) using the hydrodynamics coefficients calculated by Prestero [25].

4.2 Path Following Controller

The path following controller of the AUV model used the integral line-of-sight (iLOS) guidance law to set the yaw and pitch angles for following the generated path. The controller enables the AUV to shape its convergence towards the planned path in the presence of ocean currents and environmental disturbance by using the iLOS guidance law [26]. The desired iLOS yaw angle (heading) ψ_d and pitch angle θ_d can be determined from Eqs. (37) and (38).

$$\psi_d(e) \triangleq \arctan\left(\frac{e + K_{i,y} e_{\text{int}}}{\Delta_y}\right), \qquad K_{i,y}, \Delta_y > 0$$
$$\dot{e}_{\text{int}} = \frac{\Delta_y e}{\left(e + K_{i,y} e_{\text{int}}\right)^2 + \Delta_y^2} \tag{37}$$

$$\theta_d(h) \triangleq \arctan\left(\frac{h + K_{i,z} h_{\text{int}}}{\Delta_z}\right), \qquad K_{i,z}, \Delta_z > 0$$
$$\dot{h}_{\text{int}} = \frac{\Delta_z h}{\left(h + K_{i,z} h_{\text{int}}\right)^2 + \Delta_z^2} \tag{38}$$

where e is the cross-track error, h is the vertical-track error, $K_{i,y}$ and $K_{i,z}$ are the integral gains, and Δ_y and Δ_z represent the look-ahead distances for iLOS heading and pitch respectively. The integral terms of cross-track error e_{int} and vertical-track error h_{int} produce non-zero ψ_d and θ_d even when the AUV is on the planned path, allowing the vehicle to counteract any effects of ocean currents with the necessary sideslip and pitch angles. The rates of integral terms $\dot{e}_{int}$ and $\dot{h}_{int}$ reduce the integral action with large cross-track and vertical-track errors (i.e., vehicle is far from the planned path).

5 Simulations

The SDEQPSO path replanner is analyzed in this section. The mission objective was to determine an optimal path that guides the AUV towards a target safely and within minimum time.

5.1 Simulation Setup

The AUV mission was simulated in 2D scenarios and subsequently 3D scenarios based on the Monte Carlo method with 1000 runs. The machine used has Intel Core i5-6300U CPU @ 2.4 GHz with 8 GB RAM. The problem spaces of the simulations were assumed to be an underwater environment that contains 1000 × 1000 square grids for 2D, and 1000 × 1000 × 1000 cube grids for 3D, with a length of 1 m for each side of the grid. A priori unknown obstacles and spatiotemporal ocean currents were simulated in the problem space. The placement of the a priori unknown obstacles was configured in such a way that they will potentially block the optimized path of the AUV. In the cases of moving obstacles, they were set to move independently in different directions at random speeds up to 0.1 m/s. The variable current field with current velocity up to 0.2 m/s was generated by applying Gaussian noise to experimental data of ocean currents. The data were obtained at Beauty Point, Tasmania, Australia by using the ADCP sensors of an Explorer AUV in the University of Tasmania during one of the AUV's open water trials for the preparation of its Antarctic expedition [27].

The AUV was configured with a default water reference velocity of 1.15 m/s. Based on the properties of the REMUS 100 AUV, the safety clearance required for obstacle avoidance was defined as 3 m. The radial distance r_d was set as 50 m while the angles φ_{max} and $\theta_{\max}$ were set to 60° and 20° respectively. The test cases for the simulation are described in Tab. 1.

Table 1: Simulation test cases

Test Case	Dimension	Sonar configuration	Obstacles	Spatiotemporal currents
1	2D	Horizontal	Stationary	Yes
2	2D	Horizontal	Moving	Yes
3	3D	Horizontal	Stationary	Yes
4	3D	Horizontal	Moving	Yes
5	3D	Vertical	Stationary	Yes
6	3D	Vertical	Moving	Yes

For each run of the simulation, the maximum number of iterations for the algorithm was set to 100 with a pre-defined stopping threshold. This means the algorithm will be iterated up to a maximum number of 100 but will be stopped whenever the difference in solutions between iterations is less than the pre-set threshold. The population size of all algorithms was set to 150 particles. The setting of algorithm parameters was based on the suggested values as discussed in Section 3. The performance of the path replanner was evaluated by comparing with two other path planners:

1. SDEQPSO-based path replanner without adaptation to ocean currents,
2. SDEQPSO-based reactive path planner with adaptation to ocean currents.

Through the Monte Carlo simulation, the robustness of the planners was assessed under scenarios with stochastic processes, i.e., random-moving obstacles and random-varying ocean currents.

5.2 Simulation Results

The solutions generated by the SDEQPSO path replanner were depicted in Fig. 4. In all test cases, the mission of the AUV was to traverse the ocean field towards the target while maintaining a safe distance with the obstacles and attempting to exploit the favorable currents that would assist the AUV motion. The vehicle was driven to surf the favorable currents and to avoid the adverse currents that would oppose the vehicle's motion.

The elapsed time of the AUV mission is represented by the color bars in Fig. 4. Colors corresponding to the elapsed time are used for the planned path and the vector field of ocean currents. The boundaries of the static obstacles (Cases 1, 3 and 5) are colored brown, whereas the boundaries of the moving obstacles (Cases 2, 4 and 6) are indicated by the trails colored according to the elapsed time. Therefore, no collision will occur if the colored path does not intersect with the brown obstacles or the obstacle trails of the same color. As the obstacles were intentionally placed to block the AUV path, the AUV must detour around obstacles in all the test cases by replanning a new path whenever the previously planned paths collide with the obstacles detected by the FLS sensor. The vehicle with horizontal sonar configuration mainly maneuvered by using yaw motion, whereas the vehicle with vertical sonar configuration mostly utilized pitch motion. The resultant paths are safe and collision-free as shown in Fig. 4. During the simulation, the AUV was able to follow the planned path closely, as shown by the executed AUV paths (black lines) that closely resemble the planned paths in all test cases.

The feasibility of the path solutions can be checked by analyzing the cross-track errors of the executed paths relative to the planned paths. The calculated cross-track errors are graphed in Fig. 5. The errors for all cases were found to be well below 1 m (less than 0.1% of the total path length), proving that the AUV was able to follow the planned paths closely.

The path solutions were then validated against the vehicular constraints of the REMUS 100. The minimum turning radius of the REMUS 100 is 8.1 m in the worst-case scenario [28] and its pitch limitation is 20° (0.349 rad) based on a conservative assumption. A feasible path must have its curvature radius greater than the AUV's minimum turning radius. As shown in Fig. 6, the paths generated by the SDEQPSO satisfied the vehicle's turning and pitching constraints, validating the feasibility of the generated paths for the REMUS 100 AUV.

Next, the performances of the path replanner were assessed and compared with two other path planners based on the following properties: solution qualities, stabilities, convergence behaviors, and computational requirements. In order to study these properties, the fitness values of the obtained solutions and the computational time required to obtain the solutions were analyzed. The fitness value of a solution is given by the time required by the AUV to arrive at the target by following the generated path. Therefore, a lower fitness value is the indicator of higher solution quality and hence, a stronger search ability.

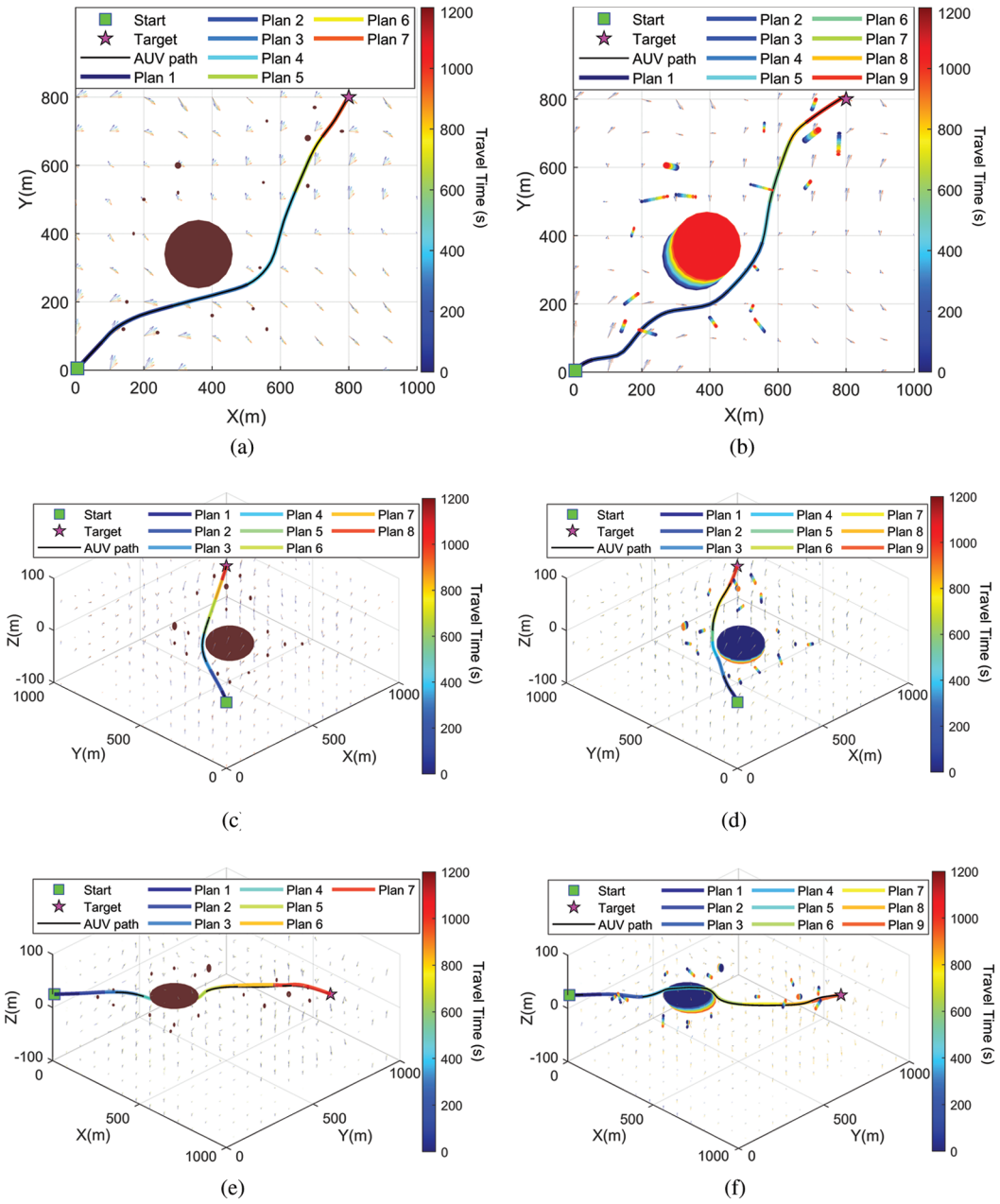


Figure 4: Pareto-optimal path solutions for (a) Case 1, (b) Case 2, (c) Case 3, (d) Case 4, (e) Case 5, and (f) Case 6

Shapiro-Wilk test with a significance level of 0.05 was used to examine the normality of the simulation results. The normality test revealed that the data was not normally distributed. Hence, medians and interquartile ranges were used as indicators for solution quality and stability. Fig. 7 shows the boxplots of the simulation results. In the boxplots, the whisker indicates the range of data. The horizontal lines inside the boxes show the medians. The upper and lower quartiles are represented by the upper and lower ends of the box, which indicates the interquartile range. The lower ends of the whiskers in the boxplot of fitness value identifies the best-known fitness for the path planners in each case.

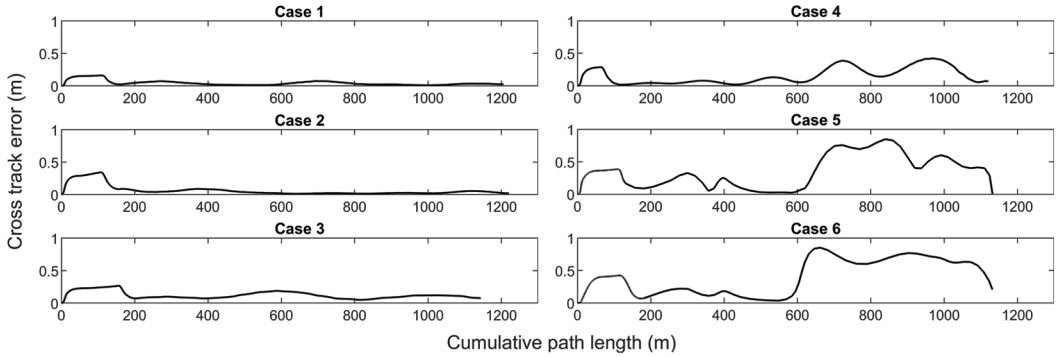


Figure 5: Variation of cross-track errors of executed paths relative to planned paths

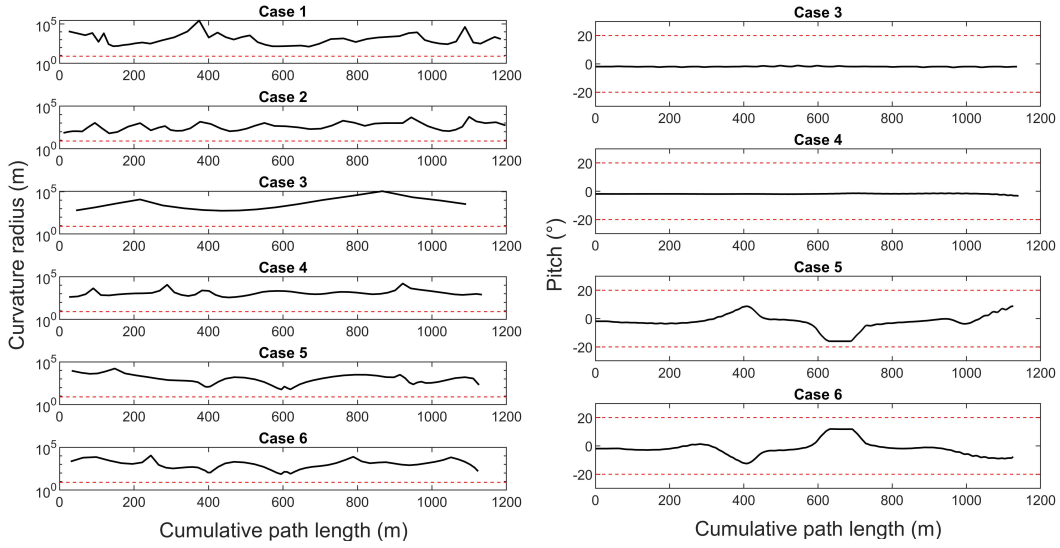


Figure 6: Variation of path curvature radius (left) and vehicle pitch (right) with respect to vehicular constraints (dashed line)

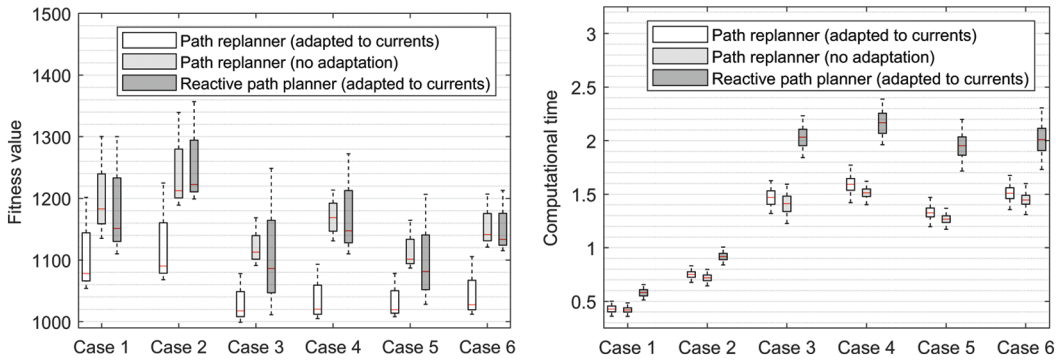


Figure 7: Fitness values (left) and computational time (right) obtained by different path planners

The effect of ocean currents on the AUV's performance was examined by comparing the proposed path replanner (adapted to currents) to the path replanner that was configured without the adaptation to currents. In contrast to the time-optimal path generated by the currents-adapted path replanner, the second replanner

simply searched for the path with the shortest distance. Fig. 7 shows that the currents-adapted path replanner generated solutions with lower medians and best-known fitness values in all test cases, suggesting a higher solution quality. The time-optimal path produced up to 13% reduction in travel time compared to the path with the shortest distance. As the shortest distance path did not take into consideration the effect of currents, the AUV that followed this path might run into adverse currents that opposed its motion and pushed it away from its path, leading to a less efficient operation. It was observed in Fig. 7 that the additional computational load for adapting the path solutions to ocean currents caused a slight increase in the computational time required by the path replanner. The simulation results showed a maximum of 5% increase in computational time, which was found to be acceptable and insignificant (less than 80 ms).

When the path replanning scheme was compared with the reactive planning, Fig. 7 shows that the medians and the best-known fitness values of the path replanner were better (lower) than the reactive planner in every test case. The path replanner was able to provide up to 11% improvement in terms of fitness values over the reactive planner, indicating the higher solution quality generated by the replanner. The path replanner was able to achieve better results because the replanning initialized the search for the optimal path from the previous solutions including the previously optimized path, whereas the reactive planner always initialized from the randomly generated solutions. This caused the reactive path planner to have poorer convergence and inadequate search before the iteration was stopped to output its final solution. In Fig. 7, it can be observed in some cases that the best-known fitness values obtained by the reactive path planner were close to the path replanner (less than 2% difference for Case 3 and Case 5). This is because the reactive path planner also used the SDEQPSO algorithm, which is a metaheuristic optimization algorithm. This means that the stochastic solutions generated by the reactive planner also have the possibility to converge at the Pareto-optimal solution although it is less likely to occur. Nonetheless, the resultant medians of fitness values produced by the reactive path planner were still worse than the replanner, leading to a significantly higher interquartile range in most test cases. The lower interquartile range of fitness values generated by the path replanner indicates higher stability and robustness in all tested scenarios.

In terms of computational time, the path replanner also outperformed the reactive path planner as shown by the shorter average time required by the replanner in all test cases. The difference in their computational time is even more significant (up to approximately 30%) when the dimension of the problem increases to 3D. The reactive path planner required longer computational time because it needs to start afresh to search for the optimal path from the randomly generated solution every time, leading to a lower rate of convergence and inefficient computation. The path replanner has faster convergence and thus shorter computational time required as a result of reusing the previous solution to effectively search for the new optimal path. The higher solution quality and shorter computational time required by the SDEQPSO path replanner indicate its higher computational efficiency. Furthermore, the consistent performance of the path replanner throughout the Monte Carlo simulation under stochastic processes verifies its robustness in generating a safe and feasible AUV path.

6 Conclusion

In this study, the SDEQPSO algorithm has successfully employed for an online path replanner of an AUV in a dynamic operational environment. Using the onboard measurements from various sensor configurations, the proposed path replanner incorporated the effect of ocean currents in path optimization to continuously generate a time-optimal path for the AUV throughout its mission. Based on the simulation results, the time-optimal path generated by the proposed path replanner offered up to 13% reduction in travel time compared to a path replanner that neglected the effect of currents. The proposed path replanning technique was also proven to have better performance over a reactive path planner in terms of solution quality (provides up to 11% improvement in fitness values), stability and computational

efficiency (provides up to 30% reduction in computational time). With the verified robustness through the Monte Carlo method, the generated path fulfilled the vehicle's safety and vehicular constraints, while taking into consideration the effect of ocean currents to improve the vehicle's operational efficiency. The future extension of this work will include incorporating noise in the sensor measurements during the simulation. Hardware-in-the-loop simulation in the AUV control software can also be applied to further evaluate the performance of the path replanner during the mission planning stage prior to actual AUV trials.

Acknowledgement: The authors acknowledge Autonomous Maritime Systems Laboratory (AMSL) in the Australian Maritime College (AMC) for providing the data from the open water trial conducted in July 2017 at Beauty Point, Tasmania, Australia.

Funding Statement: The present work is supported by a Tasmania Graduate Research Scholarship provided by AMC, University of Tasmania.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Huynh, V. T., Dunbabin, M., Smith, R. N. (2015). Predictive motion planning for AUVs subject to strong time-varying currents and forecasting uncertainties. *IEEE International Conference on Robotics and Automation,* Seattle, WA, 1144–1151.
2. Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., Neil, B. (2020). Particle swarm optimization algorithms with selective differential evolution for AUV path planning. *International Journal of Robotics and Automation, 9(2),* 94–112. DOI 10.11591/ijra.v9i2.pp94-112.
3. Zeng, Z., Sammut, K., Lian, L., He, F., Lammas, A. et al. (2016). A comparison of optimization techniques for AUV path planning in environments with ocean currents. *Robotics and Autonomous Systems, 82,* 61–72. DOI 10.1016/j.robot.2016.03.011.
4. Youakim, D., Ridao, P. (2018). Motion planning survey for autonomous mobile manipulators underwater manipulator case study. *Robotics and Autonomous Systems, 107,* 20–44. DOI 10.1016/j.robot.2018.05.006.
5. Kruger, D., Stolkin, R., Blum, A., Briganti, J. (2007). Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. *Proceedings IEEE International Conference on Robotics and Automation,* Roma, Italy, 4265–4270.
6. Koay, T. B., Chitre, M. (2013). Energy-efficient path planning for fully propelled AUVs in congested coastal waters. *MTS/IEEE OCEANS,* Bergen, Norway, 1–9.
7. Sun, B., Zhu, D. (2016). Three dimensional D∗ Lite path planning for autonomous underwater vehicle under partly unknown environment. *12th World Congress on Intelligent Control and Automation,* Guilin, China, 3248–3252.
8. Rao, D., Williams, S. B. (2009). Large-scale path planning for underwater gliders in ocean currents. *Australasian Conference on Robotics and Automation,* Sydney, Australia, 2–4.
9. Hernández, J. D., Vidal, E., Moll, M., Palomeras, N., Carreras, M. et al. (2019). Online motion planning for unexplored underwater environments using autonomous underwater vehicles. *Journal of Field Robotics, 36(2),* 370–396. DOI 10.1002/rob.21827.
10. Alvarez, A., Caiti, A., Onken, R. (2004). Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering, 29(2),* 418–429. DOI 10.1109/JOE.2004.827837.
11. MahmoudZadeh, S., Yazdani, A. M., Sammut, K., Powers, D. M. (2018). Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms. *Applied Soft Computing, 70,* 929–945. DOI 10.1016/j.asoc.2017.10.025.
12. Lim, H. S., Fan, S., Chin, C. K. H., Chai, S. (2018). Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner. *IEEE OES Autonomous Underwater Vehicle Symposium,* Porto, Portugal, 1–7.

13. Fu, Y., Ding, M., Zhou, C., Hu, H. (2013). Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(6),* 1451–1465. DOI 10.1109/TSMC.2013.2248146.
14. Lolla, T., Ueckermann, M., Yiğit, K., Haley, P. J., Lermusiaux, P. F. (2012). Path planning in time dependent flow fields using level set methods. *2012 IEEE International Conference on Robotics and Automation,* Saint Paul, MN, 166–173.
15. Witt, J., Dunbabin, M. (2008). Go with the flow: optimal AUV path planning in coastal environments. *Australasian Conference on Robotics and Automation,* Canberra, Australia, 1–9.
16. Yao, X., Wang, X., Wang, F., Zhang, L. (2020). Path following based on waypoints and real-time obstacle avoidance control of an autonomous underwater vehicle. *Sensors, 20(3),* 795. DOI 10.3390/s20030795.
17. Sun, B., Zhu, D., Yang, S. X. (2018). An optimized fuzzy control algorithm for three-dimensional AUV path planning. *International Journal of Fuzzy Systems, 20(2),* 597–610. DOI 10.1007/s40815-017-0403-1.
18. Zeng, Z., Sammut, K., Lammas, A., He, F., Tang, Y. (2015). Efficient path re-planning for AUVs operating in spatiotemporal currents. *Journal of Intelligent & Robotic Systems, 79(1),* 135–153. DOI 10.1007/s10846-014-0104-z.
19. Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M. et al. (2015). Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. *Journal of Field Robotics, 32(7),* 952–983. DOI 10.1002/rob.21554.
20. Piegl, L., Tiller, W. (2012). *The NURBS book*. Berlin: Springer Science & Business Media.
21. Garau, B., Alvarez, A., Oliver, G. (2006). AUV navigation through turbulent ocean environments supported by onboard H-ADCP. *Proceedings 2006 IEEE International Conference on Robotics and Automation,* Orlando, FL, 3556–3561.
22. Sun, J., Lai, C. H., Wu, X. J. (2012). *Particle swarm optimisation classical and quantum perspectives*. Boca Raton, FL: CRC Press.
23. Lim, H. S., Fan, S., Chin, C. K. H., Chai, S., Bose, N. et al. (2019). Constrained path planning of autonomous underwater vehicle using selectively-hybridized particle swarm optimization algorithms. *IFAC-PapersOnLine, 52(21),* 315–322. DOI 10.1016/j.ifacol.2019.12.326.
24. Fossen, T. I. (1999). *Guidance and control of ocean vehicles*. Norway: John Wiley & Sons.
25. Prestero, T. T. J. (2001). *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle*. Cambridge, MA: Massachusetts Institute of Technology.
26. Caharija, W., Pettersen, K. Y., Gravdahl, J. T., Børhaug, E. (2012). Path following of underactuated autonomous underwater vehicles in the presence of ocean currents. *IEEE Conference on Decision and Control,* Maui, HI, 528–535.
27. Pyper, W. (2018). Yellow submarine prepares for first Antarctic mission. *Australian Antarctic Magazine,* 12–13.
28. Eng, Y., Teo, K. M., Chitre, M., Ming Ng, K. (2016). Online system identification of an autonomous underwater vehicle via in-field experiments. *IEEE Journal of Oceanic Engineering, 41(1),* 5–17.

Real-time implementation of an online path replanner for an AUV operating in a dynamic and unexplored environment

Hui Sheng Lim [a], Peter King [a], Christopher K.H. Chin [a], Shuhong Chai [a] and Neil Bose [a,b]

[a] National Centre for Maritime Engineering and Hydrodynamics, Australian Maritime College, University of Tasmania, Launceston, TAS, 7250, Australia

[b] Memorial University of Newfoundland, St. John's, NL A1C 5S7, Canada

Abstract

This study describes the implementation of an online path planner in an autonomous underwater vehicle (AUV) system by using an open-source system architecture, MOOS-IvP. The path planner employed a path replanning scheme and the selective differential evolution quantum-behaved particle swarm optimization (SDEQPSO) algorithm. The implementation was based on a modular framework to ensure the robustness of the path replanner during a mission. The performance of the path replanner was evaluated and verified under stochastic processes in hardware-in-the-loop (HIL) tests, in which the replanner interacted with the onboard controllers and actuators of an Explorer AUV. The experimental results showed the path replanner can be run seamlessly with the hardware onboard an Explorer AUV in real time to generate and continuously refine a safe and feasible path for a dynamic and unexplored environment.

Keywords: Autonomous underwater vehicle; path planning; particle swarm optimization; hardware-in-the-loop simulation; real-time systems

1 Introduction

AUVs are effective tools for performing underwater operations in various industry sectors, ranging from scientific research and commercial applications to defence and security industries. As the market demands AUVs to expand their range of applications, the vehicles are required to execute increasingly challenging missions in more dynamic and constrained environments over extended durations. The energy requirements of these missions have outpaced the technology development of AUVs' onboard battery capacities [1]. As typical AUVs have limited onboard resources, efficient motion and path planning becomes one of the key factors for completing missions that challenge the vehicles' autonomy and endurance.

A path planner can improve the performance and endurance of an AUV by exploiting ocean currents [2]. Spatiotemporal currents, which are present in any AUV missions, can have a significant impact on the vehicle's performance. Strong ocean currents and eddies can oppose the vehicle motions and even push the vehicle off its planned path, leading to an increase in its energy consumption and thus a reduction in its endurance. A path planner that considers the effect of ocean currents can generate a time-optimal path, which increases the operational efficiency of an AUV by guiding the vehicle towards its target within minimum time [3]. A time-optimal path planner is adapted to its surrounding current profile. This enables an AUV to surf the favourable currents that can assist its motion while preventing it from running into the adverse currents that can oppose it. A time-optimal path is particularly important when the AUV is required to traverse between multiple regions of interest across large bodies of water and over an extended duration.

Due to the spatial and temporal variabilities of ocean environments, in which an AUV may encounter moving obstacles and time-varying ocean currents, a pre-planned path may become less optimal or even physically infeasible over time [4]. The vehicle may also deviate significantly from the planned path due to current disturbance. These problems are most notably evident for AUVs in missions with extended durations. To ensure a safe and optimal path for an AUV operating in dynamic environments, path planning needs to be carried out online and continuously throughout the AUV's mission. This study aims to improve the performance of an AUV operating in a dynamic and unexplored environment by implementing an online path planner that can generate time-optimal paths based on real-time vehicular and environmental feedback.

1.1 Background and Related Work

Online path planning pertains to the process where a path is continuously planned and refined in real time for a vehicle subjected to dynamic variabilities. An online path planner enables a vehicle to adapt to unexpected changes in a dynamic and/or unknown environment, which is a common operational scenario for an AUV. The main challenge of online path planning is finding an optimal path within a limited time allowed for planning. A practical online path planner should establish a balance between its computational load and the quality of generated paths.

An intuitive approach to online planning is known as local path planning, in which local adjustments are made for a vehicle while following a previously planned path. Local path planning mainly concerns avoiding collisions with an obstacle and guiding the vehicle back to the planned path after passing the obstacle. Previous studies [5-7] integrated obstacles avoidance control with path following to cope with a dynamic environment effectively but at the expense of path optimality. Based on a similar approach, Sun, Zhu [8] improved the quality of generated paths by using fuzzy control and the quantum-behaved particle swarm optimization (QPSO) algorithm. Benjamin, Defilippo [9] proposed a dynamic obstacle manager that uses multi-objective optimization of interval programming to ensure a collision-free trajectory while following a pre-planned path.

Some studies adopted a reactive path planning approach, in which a new path is generated to react to a dynamic environment after discarding previously planned paths. Existing studies proposed to combine the reactive approach with various algorithms such as Voronoi diagram [10], artificial potential fields (APF) [11, 12], A* [13-15], Field D* [16], and rapidly-exploring random trees (RRT) [17, 18]. Although the APF method is fast and suitable for high-dimensional problems, it is very susceptible to local minima. Search-based methods, such as A* and Field D*, are low complexity algorithms that can only be applied to lower-dimensional and less complex problems. RRT is a widely applied sampling-based method that is effective for high-dimensional and highly time-constraint scenarios, but it generates suboptimal paths that require further refinement. Belkhouche, et al. [19, 20] also proposed reactive path planners that use simple collision cones and kinematic-based navigation laws to generate suboptimal but safe and robust paths. Reactive planning approach was also applied with neural network and reinforcement learning to generate safe AUV paths in dynamic environments [21-23]. Cheng and Zhang [24] proposed a deep reinforcement learning algorithm that combined multiple reward functions to achieve effective obstacle avoidance in unknown environments, but it did not demonstrate significant advantages over non-learning algorithms. Although these learning algorithms showed their effectiveness in numerical simulations, their implementation in AUVs are challenging because model training for an AUV in real world is expensive and time-consuming. Training a model in simulations is possible but the model will become biased and very specific to the simulated environments. None of the abovementioned path planners was verified experimentally on an actual AUV platform.

Contrary to reactive path planning, path replanning is a technique that generates a path by reusing and modifying the previous solution(s) instead of starting afresh in every planning cycle. Usually in oceans, the environmental changes occur gradually rather than transform completely and abruptly. The planning conditions for a new path may bear some resemblance to the conditions of the previous planning cycle. Hence, generating a new path based on the previously planned path is considered more computational efficient because it is probable that the new path nodes can be located in proximity to the previous solution(s). The computational efficiency gained by reusing previous solution(s) can be profound especially for missions that involve highly dynamic and large operational environments.

Previous studies proposed path replanning based on a single previous solution. For example, an “anytime” approach was developed to generate a feasible but suboptimal path that can be modified and refined continuously throughout the mission. This approach can be applied with A* [25], Field D* [26] and RRT [27]. Park, Pan [28] adopted a similar approach and developed a path replanner that uses parallel computing on multi-core processors to improve its computational performance. Other algorithms were also used to replan a new path from a previous solution, such as the stochastic trajectory optimization motion planning (STOMP) [29], D* lite [30], and RRT [31, 32]. Path replanning can be also be conducted by reusing more than one previous solution. Bruce and Veloso [33] developed an RRT-based path replanner that stores the cache of all the previous waypoints for path replanning. To enable path replanning from multiple solutions, existing studies also proposed the use of homotopic sets of paths with APF [34] and RRT [35].

Path replanning based on a pool of previous solutions can also be achieved by employing a population-based metaheuristic optimization algorithm, which can fully preserve its previous solutions throughout a mission. For example, MahmoudZadeh, Yazdani [36] successfully applied differential evolution (DE), particle swarm optimization (PSO), firefly algorithm (FA) and biogeography-based optimization (BBO) for path replanning. The

application of PSO in path replanning was explored by several recent studies [37-39]. Zeng, Sammut [40] also proposed a path replanner that used the QPSO algorithm to replan an AUV path at a preset fixed interval. This path replanner is only applicable for an AUV equipped with a high-end sensor setup because its planning process requires the vehicle to provide all environmental information encircling the vehicle up to a specific radius.

Although the PSO algorithm and its variants offer an effective and computationally efficient solution for online path planning, there are concerns about the practicability of PSO-based path planners such as the convergence of PSO at local minima due to limited time for online planning, as well as their computational loads when implemented in an actual vehicle. Based on the findings of recent studies [41-43], Lim, Fan [44] proposed the SDEQPSO algorithm that uses the selective hybridization of DE to enhance the swarm's searching ability and resistance to local minima without inflating the computational load. The SDEQPSO algorithm showed higher computational efficiency and better path planning performance than RRT* and several evolutionary algorithms, such as the standard PSO, QPSO, DE, FA and genetic algorithm (GA). Lim, Chin [45] also developed a path replanner by using the SDEQPSO algorithm to solve the online path planning problem of the REMUS 100 AUV in numerical simulations [45].

Despite the extensive research on AUV path planning techniques, the majority of online path planners, especially sophisticated planners, were verified only in pure numerical simulations due to technical challenges and costs associated with implementation in an actual AUV. Only a small number of the abovementioned path planners were tested experimentally or implemented in a real-time system [9, 12, 21, 29, 32, 35]. Numerical verification is insufficient because it often does not consider an AUV's actual capability to compute the paths and execute them in real time. It is crucially important to assess the performance and robustness of a path planner when implemented in a real-time vehicle system. In particular, an online path planner must be examined for its capability in handling real-time feedback from the vehicle and its surrounding environments.

In this study, a novel online path replanner was developed and implemented in an actual AUV platform. The path replanner used the SDEQPSO algorithm, which was proposed in the preliminary studies [44, 45]. This study addresses the implementation and verification of the SDEQPSO path replanner in a real-time vehicle system. It extends from the previous work by offering several algorithmic improvements and further experimental verifications. In particular, the work improved the robustness and stability of the replanner in real-time systems by building the algorithm as an independent application in a modular framework. The objective functions of the replanner were refined to address the concerns for uncertainties in sensor measurements, as well as the uncertainties arisen from vehicular actuation and external disturbance in operational environments. Most importantly, unlike the pure numerical work in the previous studies, the presented experimental results verified the replanner by examining its interaction with the hardware of an Explorer AUV in experiments that involved various sensor configurations and test scenarios. This study contributes to the development of a real-time testing framework for implementing an online path planner. A fast and efficient novel evolutionary algorithm that is suitable for real-time planning was developed. In addition, the proposed path replanner offers the following advantages:

- It can generate a time-optimal path that exploits ocean currents to improve the performance of an AUV.
- It continuously adapts a path to the spatiotemporal variabilities of an ocean environment and the constraints imposed by an AUV and its missions.
- It does not require pre-planned paths, system models or any prior knowledge of the terrain/environment.
- It is scalable for missions that require different sensor configurations.
- It has the versatility to accept different current profile data for the generation of time-optimal paths.
- Its implementation is based on a modular framework in an open-source system, which promotes ease of applications and extendibility for different robotic platforms.

The rest of this paper is organised as follows. Section 2 explains the framework used to implement the path replanner. In Section 3, the SDEQPSO algorithm and the proposed path replanning scheme are described. Section 4 formulates the path replanner and its objective functions. Section 5 describes the vehicle and sensors models used in the experiment. The experimental setup, results and discussion are presented in Section 6. This study is concluded in Section 7 with a summary and directions for future work.

2 Hardware-in-the-loop Test Framework

A hardware-in-the-loop (HIL) test is a technique used for developing and testing real-time embedded systems and algorithms. A HIL test of an AUV can be conducted on dry land without requiring the deployment of the vehicle

in water, hence reducing the operational cost. It can serve as a platform for effective prototyping of AUV control algorithms in a safe and controlled environment before deploying the algorithm for actual field operations. Implementation of a path planner in a HIL test provides insight into how the planner interacts with the high-level and low-level controllers of an AUV in real time.

During the HIL test of an AUV, an interface is required to enable information exchange between the actual vehicle system and the algorithm under test. The Mission Oriented Operating Suite (MOOS) is a middleware that can serve as an interface for a HIL test. In the industry of marine robotics, the MOOS framework is commonly applied for field operations because of its wide-ranging capabilities and platform independence [46]. Therefore, MOOS is often used for HIL tests as well as numerical simulations to allow for an easy transition into field operations.

MOOS [46] is an open-source and cross-platform middleware that adheres to the ISO (ANSI) C++ standard to ensure platform independence. Based on a publish-subscribe architecture, the MOOS middleware functions as a centralized database to enable information exchange between a community of vehicle system processes, which are run independently as MOOS applications. The MOOS library contains a collection of essential applications for robotic operations, ranging from autonomy, sensor management, and communication to debugging and data postprocessing. This reduces the time required to prototype a control algorithm when the MOOS software is used. MOOS-IvP [47] is a comprehensive marine autonomy suite, in which the MOOS software is complemented by additional MOOS applications, including the IvP helm. The IvP helm uses interval programming (IvP) and a behaviour-based architecture to solve the multi-objective optimization of competing behaviours in a robotic system. The IvP helm represents each objective function as a behaviour that uses a piecewise linear approximation. As a single MOOS application, the IvP helm controls the desired speed, depth, and orientations of a vehicle by arbitrating multiple behaviours. The MOOS-IvP software was successfully applied for algorithm implementations in several recent studies [4, 9, 48-50].

The SDEQPSO path replanner was implemented using the MOOS-IvP framework and analysed in a real-time HIL test. The MOOS-IvP framework is modular and uses a kernel that runs all MOOS applications independently to ensure the robustness of the system. The HIL test was set up using a backseat driver paradigm as described in Figure 1 and Table 1. The backseat driver separates the autonomy of an AUV from its vehicle control. The frontseat controller of the vehicle onboard computer executes the vehicle control, while the payload computer in the backseat is responsible for the vehicle autonomy.

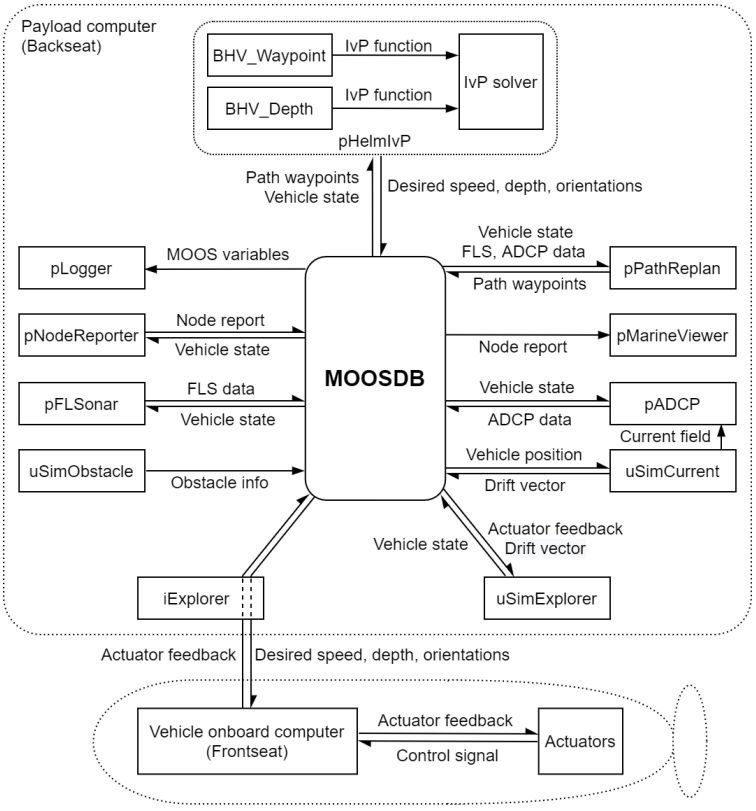


Figure 1: Backseat driver paradigm of HIL test using MOOS-IvP.

Table 1: Descriptions of components in the HIL test.

Component	Type	Description
MOOSDB	MOOS database	A centralized database for all MOOS applications.
pPathReplan	MOOS application	Implementation of the SDEQPSO path replanner.
pNodeReporter	MOOS application	Generate a node report that consists of the vehicle information, including its position, speed, heading, etc.
pMarineViewer	MOOS application	Receive the node report to produce a visualization of vehicles and associated information.
pLogger	MOOS application	Record all variables sent between applications during a test for post-mission analysis, data gathering and post-mission replay.
uSimExplorer	MOOS application	A 3D vehicle simulator of an Explorer AUV.
iExplorer	MOOS application	Establish serial communications between the payload computer and the onboard computer.
uSimObstacle	MOOS application	Simulate unknown and dynamic obstacles in the problem space.
pFLSonar	MOOS application	Simulate the data detected by an FLS sensor based on the vehicle position, orientations and obstacle information.
uSimCurrent	MOOS application	Simulate time-varying ocean currents in the problem space.
pADCP	MOOS application	Simulate the data of a current field inferred by an H-ADCP sensor.
pHelmIvP	MOOS application	IvP helm for arbitrating multiple behaviours through an IvP solver.
BHV_Waypoint	IvP behaviour	Path following behaviour for traversing a sequence of waypoints.
BHV_Depth	IvP behaviour	Depth control behaviour for tracking a sequence of specified depths.
Vehicle computer	Hardware	Generate control signals based on the desired speed, depth and orientations.
Actuators	Hardware	Actuate the control surfaces and propeller based on the control signals.

During the experiment, the physical actuators of the AUV were controlled directly by the vehicle's embedded computer, while the hull remained stationary on dry land. The problem space was a virtual ocean environment, which was simulated by the uSimCurrent and uSimObstacle applications. The motion response of the AUV was modelled in the test by using the uSimExplorer application, which is a simulation model of an Explorer AUV. The AUV was equipped with a forward-looking sonar (FLS) and a horizontal acoustic Doppler current profiler (H-ADCP). The vehicle and sensor models used in the HIL test are described in Section 5.

3 SDEQPSO Path Replanner

The SDEQPSO algorithm belongs to the QPSO family. The algorithm contains quantum-behaved particles that move in multidimensional space to determine the optimal solution from a pool of potential solutions. The solutions are recorded by the positions of particles. For SDEQPSO with N particles in a D-dimensional space of an objective function f, the position vector of the current i^{th} particle at t^{th} iteration can be given as follows:

$$X_i^t = \left[\chi_{i1}^t, \chi_{i2}^t, ..., \chi_{ij}^t, ..., \chi_{iD}^t \right], \quad i \in \{1, 2, ..., N\} \tag{1}$$

In SDEQPSO, each dimension of the particle's position is attracted by a quantum potential well centred on a local attractor. The quantum state of the particle is described by a wave function. The position of the particle can be measured from the probability distribution function of the wave function by using Monte Carlo inverse transformation [51]. This process collapses the quantum state of the particle to a classical state to produce the following equation for defining its position.

$$X_i^{t+1} = \begin{cases} \delta_i^t + 0.5 \cdot L_i^t \cdot -\ln\left(u_i^t\right), & \text{if } u \geq 0.5 \\ \delta_i^t - 0.5 \cdot L_i^t \cdot -\ln\left(u_i^t\right), & \text{if } u < 0.5 \end{cases} \tag{2}$$

where u is a vector of random number uniformly distributed in the range of 0 – 1. The δ potential well is defined by Equation (3), while L is the characteristic length of the δ potential well as given in Equation (4). δ and L can be determined by the particle's personal best position (*pbest*), the swarm's mean best position (*mbest*) and the swarm's global best position (*gbest*) as given in Equation (5), (6) and (7), respectively.

$$\delta_i^t = \phi_i^t \cdot pbest_i^t + \left(1 - \phi_i^t\right) \cdot gbest^t \tag{3}$$

$$L_i^t = 2 \cdot \beta \cdot \left| mbest^t - X_i^t \right| \tag{4}$$

$$pbest_i^t = \begin{cases} pbest_i^{t-1}, & \text{if } f(X_i^t) \geq f(pbest_i^{t-1}) \\ X_i^t, & \text{if } f(X_i^t) < f(pbest_i^{t-1}) \end{cases} \tag{5}$$

$$mbest^t = \sum_{i=1}^{N} \frac{pbest_i^t}{N} \tag{6}$$

$$gbest^t = \arg\min \left[f(pbest_i^t) \right] \tag{7}$$

where ϕ in Equation (3) is a vector of random number uniformly distributed in the range of 0 – 1. The parameter β in Equation (4) is a positive real number known as the contraction-expansion (CE) coefficient. By merging Equation (2) - (7), the following position update equation of particles can be obtained.

$$X_i^{t+1} = \phi_i^t \cdot pbest_i^t + \left(1 - \phi_i^t\right) \cdot gbest^t \pm \beta \cdot \left| mbest^t - X_i^t \right| \cdot -\ln\left(u_i^t\right) \tag{8}$$

The CE coefficient β can be adjusted to control the global and local search of the algorithm. Based on the recommendation in the empirical study of QPSO [51], a common strategy is to set β at an initial value $\beta_{\max}$ of 1.0 and linearly decrease it to a minimum value $\beta_{\min}$ of 0.5 using Equation (9) as the algorithm iterates.

$$\beta = \beta_{\max} - \frac{t}{t_{\max}} (\beta_{\max} - \beta_{\min}) \tag{9}$$

The SDEQPSO executes the DE operation on the particles based on a selective scheme. The selective hybridization of DE improves the diversity of the particle swarm and thus its searching ability while maintaining the swarm dynamics. After updating the positions of particles, all particles are sorted based on their personal best positions. Next, a number of particles are selected to go through the DE operation. A selection factor S can be used to adjust the number of selected particles N_S according to Equation (10).

$$N_S = N \times S, \quad S \in [0,1] \tag{10}$$

The three-step DE operation consists of the mutation, crossover and selection operators. The N_S best-performing particles undergo mutation and crossover to generate trial vectors, which replace the N_S worst-performing particles by natural selection.

1. A donor vector U is produced by mutation based on Equation (11).

$$U_i^t = gbest^t + \frac{(pbest_{r1}^t - pbest_{r2}^t) + (pbest_{r3}^t - pbest_{r4}^t)}{2} \tag{11}$$

   where r_1, r_2, r_3 and r_4 are randomly selected particle indices that are mutually different, and different from the selected particle index i and the global best particle index, i.e., $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i \neq gbest$.

2. A trial vector T is produced by crossover between the donor vector and the selected particle's personal best position based on Equation (12).

$$\begin{aligned} T_i^t &= \left[\tau_{i1}^t, ..., \tau_{ij}^t, ..., \tau_{iD}^t \right] \\ \tau_{ij}^t &= \begin{cases} u_{ij}^t, & \text{if } r_j \leq CR \,||\, j = r \\ pbest_{ij}^t, & \text{if } r_j > CR \,||\, j \neq r \end{cases} \end{aligned} \tag{12}$$

where CR denotes the crossover probability that has a recommended value of 0.85. r_j is a random number uniformly distributed in the range of 0 – 1. r is a random integer in the range of 1 to the number of particle dimensions D.

3. Natural selection is performed to replace the worst-performing particle with the trial vector T. Only the worst-performing particles are replaced so all potentially best solutions can be retained.

For path planning problems, the selection factor S has a recommended setting of 0.3. This setting promotes swarm diversity while maintaining an adequate number of potentially optimal particles [44]. By increasing the diversity and evolutionary rate of the swarm through the removal of the least desirable solutions, the selective DE operation in SDEQPSO can achieve a better and faster global convergence.

The SDEQPSO path replanner can continuously generate and refine an AUV path at an adaptive interval throughout a mission. The adaptive replanning interval was devised to react to environmental changes, which can cause a previously planned path to become unsafe or less optimal. The following flags can trigger path replanning:

- Elapsed time after a previous planning reaches a predefined limit.
- Newly detected obstacles show up in an AUV's safe zone.
- Newly detected obstacles conflict with the previously planned path.

In order to ensure the planned path was optimized for a spatiotemporal current field, the first replanning flag enabled the path replanner to refine the path periodically even when no conflicting obstacles were detected. The elapsed time limit was defined to be inversely proportional to the vehicle and current velocities. Based on the vehicle water-referenced velocity V_a and the maximum current velocity $V_{c,max}$ measured by a current profiler, the elapsed time limit t_{replan} can be given as below:

$$t_{replan} = \frac{D_{adcp}}{V_a + V_{c,max}} \tag{13}$$

where D_{adcp} is the effective range of the vehicle's current profiler.

When replanning is required, the SDEQPSO path replanner can use the solutions from its previous planning cycle as the initial solutions for planning a new path that is optimized for its new planning conditions. Figure 2 describes the path replanning process. The black circles and grey dots in Figure 2 depict the potential path nodes that can form an AUV path. During a new planning cycle, some of the previous solutions (grey dots) can be preserved and reused as the initial solutions (black circles) for path optimization.

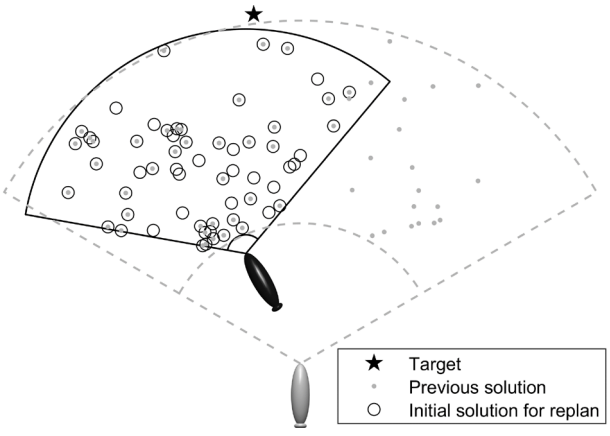


Figure 2: Reusing previous solutions for path replanning (black vehicle depicts the current state and grey vehicle depicts the previous state).

The initialisation of path replanning begins with finding the new boundary conditions based on a new starting point, i.e., the current position of the vehicle. Path nodes behind the new starting point are deleted. Solutions within the new boundary conditions are preserved, while solutions outside the boundary are rejected and regenerated. After the initialisation process, the solutions undergo the iteration of SDEQPSO to find the optimal path. The following pseudocode describes the algorithm flow of the SDEQPSO path replanner.

Step 1. **Define** the parameters of the algorithm and operational environments.
Step 2. **Initialise** random particle positions in Equation (1). Set *pbest* of particles to their current positions.
Step 3. **Check** the path replanning flag.
Step 4. **If** replanning flag == TRUE
While the stop criteria are not met,
For $t = 1, 2, \ldots, t_{max}$,
Evaluate the fitness $F(X_i^t)$ of particles by using the objective function F.
Update *pbest* and *gbest* by using Equations (5) and (7) respectively.
Update *mbest* by using Equation (6).
Update β by using Equation (9).
For each particle $i = 1, 2, \ldots, N$,
Update the positions of particles by using Equation (8).
End
Sort all particles according to their personal best fitness.
For $k = 1, 2, \ldots, N_s^{th}$ best performing particle,
Mutation to generate a donor vector U_k^t by using Equation (11).
Crossover to generate a trial vector T_k^t by using Equation (12).
Natural selection to replace k^{th} worst-performing particle with T_k^t.
End for
End for
End while
Output *gbest* that corresponds to the optimal path.
Else
Follow the path generated by the previous planning cycle.
End if
Step 5. **Return** to Step 3 if the target is not reached.

4 Path Formulation

In this study, the path replanner aimed to solve a multi-objective optimization problem, which required finding the time-optimal path from a group of potential paths that can guide an AUV towards its target across a dynamic and unknown ocean environment. A potential AUV path can comprise a sequence of path nodes that connect a starting point and an end point (target). Manipulating the locations of path nodes can refine and optimize a path. Neither the starting point nor the end point of a path was involved in the optimization process because all potential paths have the same start and end locations.

In the SDEQPSO path replanner, a potential path solution can be modelled as a particle in the swarm. The swarm population can be represented by a matrix $X = [X_1, X_2, \ldots, X_N]^T$, where X_i is a particle's position vector and N is the population size of the swarm. The entries in the position vector of a particle represent the coordinates of its path nodes. Assuming each path comprises n+2 nodes (including the starting and end points), a total number of n nodes is involved in the path optimization process. To register the polar coordinates of n node(s) in a 2D Euclidean plane, a particle requires a position vector with $2n$ dimensions for radial coordinates r and azimuthal angle coordinates φ. To register the spherical coordinates of n node(s) in a 3D Euclidean space, a particle requires $3n$ dimensions to include an additional n dimension(s) for polar angle coordinates θ. The position vector X of the i^{th} particle in 2D and 3D can be given as follows.

$$
\begin{aligned}
X_i^t &= \left[r_i^t, \theta_i^t \right]_{2D} \qquad\qquad i \in \{1, 2, ..., N\} \\
X_i^t &= \left[r_i^t, \theta_i^t, \varphi_i^t \right]_{3D} \\
r_i^t &= \left[\chi_{i,1}^t, ..., \chi_{i,j}^t, ..., \chi_{i,n}^t \right] \\
\theta_i^t &= \left[\chi_{i,n+1}^t, ..., \chi_{i,n+j}^t, ..., \chi_{i,2n}^t \right] \\
\varphi_i^t &= \left[\chi_{i,2n+1}^t, ..., \chi_{i,2n+j}^t, ..., \chi_{i,3n}^t \right]
\end{aligned}
\tag{14}
$$

The Cartesian coordinates of the j^{th} path node recorded by the i^{th} particle can be calculated from the polar coordinates using Equation (15) and from the spherical coordinates using Equation (16).

$$x_{i,j} = r_{i,j} \cos \varphi_{i,n+j}$$
$$y_{i,j} = r_{i,j} \sin \varphi_{i,n+j} \tag{15}$$

$$x_{i,j} = r_{i,j} \cos \varphi_{i,n+j} \sin \theta_{i,2n+j}$$
$$y_{i,j} = r_{i,j} \sin \varphi_{i,n+j} \sin \theta_{i,2n+j} \tag{16}$$
$$z_{i,j} = r_{i,j} \cos \theta_{i,2n+j}$$

To enhance the search efficiency of the path replanner, a radial boundary condition was used to control the placement of path nodes. Every path node was constrained to lie within a concentric annulus, which is the area between two concentric circles of different radii. These radii can be given by the lower boundary $R_{\min}$ and upper boundary $R_{\max}$ in Equation (17). A hard constraint was used to restrict the search domains of radial coordinates between $R_{\min}$ and $R_{\max}$.

$$R_{\min} = \left[0, r_d, 2r_d, \ \ldots \ , r_{target} \right]$$
$$R_{\max} = \left[r_d, 2r_d, 3r_d, \ \ldots \ , r_{target} \right] \tag{17}$$

where r_d denotes the radial distance between every pair of concentric circles and r_{target} is the radial coordinate of the target. A path solution is deemed infeasible and needs to be regenerated if any of its path nodes exceed the boundaries $R_{\min}$ or $R_{\max}$. By adjusting r_d, the total number of path nodes n can be manipulated according to Equation (18).

$$n = \left\lceil \frac{r_{target}}{r_d} \right\rceil \tag{18}$$

To ensure path solutions comply with the minimum turning radius and pitch limitation of an AUV, the search domain of azimuthal angle coordinates and polar angle coordinates were hard-constrained by using an azimuthal boundary φ_{max} and a polar boundary θ_{max}, respectively. A feasible path solution must have all its path nodes fulfilling the conditions of $|\varphi_{i,j}| < \varphi_{max}/2$ and $|\theta_{i,j}| < \theta_{max}/2$; otherwise, the solution must be regenerated.

The path nodes, including the starting and end points, can be connected to form an AUV path by using B-splines, which are parametric curves generated from a series of connected piecewise polynomials [52]. B-splines are suitable for this application because it offers the following properties:

- It can produce a practical path shape without unnecessary wiggling segments, hence reducing the control actions required by an AUV to follow the path.
- It can maintain the continuity of its second derivative and curvature function to ensure a smooth path. This enhances the path following performance of an AUV.
- It can offer local control for path alteration without loss of continuity. This allows the effect of path nodes relocation to be localized to the adjacent path segments only.

Path nodes can serve as the control points for a B-spline path based on the curve function in Equation (19), which produces an output vector $P(u)$ to represent a k+1 order curve in the form of discretised waypoints. If a path has n+2 nodes, the number of generated piecewise polynomials is n+1.

$$P(u) = \sum_{i=0}^{n+1} x_i \ B_{i,k}(u), \qquad i \in \{0,1,2,...,n+1\} \tag{19}$$

where x_i represents the control points, u is the non-decreasing knot sequence given by a knot vector $U = [u_0, \ldots, u_i, \ldots, u_{n+k+2}]$, and $B_{i,k}(u)$ denotes the basis functions of k degree as given by the following Cox de Boor recursion [52].

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \tag{21}$$

4.1 Objective Function

When applying PSO-based algorithms, developing suitable objective functions is essential for evaluating the fitness of particles. Fitness evaluation using objective functions normally contributes to the majority of algorithm runtime [51]. Objective functions must consider the optimization criteria and physical conditions of its problem space to accurately measure the fitness of particles. As a minimization problem, path planning needs to minimise the travel time of a vehicle. Thus, a better path solution should have a lower cost/fitness. This study used a single aggregate objective function to model the multi-objective path planning problem. Equal weights were used for the underlying objective functions F_k. The optimal path solution X^* can be defined by Equation (22).

$$X^* = \arg\min \sum_{k=1}^{2} F_k(X_i) \tag{22}$$

The primary objective function F_1 served to measure the fitness of a path based on its travel time required to reach the target while taking into account the effect of ocean currents. This function can adapt a path to ocean currents, resulting in a time-optimal path that can guide an AUV towards its target within a minimum time. A B-spline path X_i can be described by a sequence of discretised waypoints $P = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$, where P is generated by Equation (19) and m is the total number of discretised waypoints. For the i^{th} particle, its travel time cost $F_1(X_i)$ can be obtained in Equation (23) by summing the total time required to travel on all the path segments that connect the waypoints in P.

$$F_1(X_i) = \sum_{j=1}^{m-1} \frac{\left\| \overrightarrow{p_{i,j} p_{i,j+1}} \right\|}{\left| V_g \right|}, \quad j \in \{1, 2, \ldots, m-1\} \tag{23}$$

where the numerator gives the Euclidean distance between two consecutive waypoints. V_g denotes the vehicle ground-referenced velocity, which is the resultant vehicle velocity under the effect of currents. V_g can be obtained in Equation (24) by summing the vehicle water-referenced velocity V_a and the effect of current velocity V_c on the vehicle, which can be given by projecting V_c in the direction of the path vector.

$$V_g = V_a + \frac{V_c \cdot \overrightarrow{p_{i,j} p_{i,j+1}}}{\left\| \overrightarrow{p_{i,j} p_{i,j+1}} \right\|} \tag{24}$$

4.2 Obstacle Avoidance

The second objective function served as a penalty function to measure a path's exposure to obstacles in terms of a penalty cost. It was found in [53] that using a hard constraint in obstacle avoidance can generate an excellent solution quality. However, a hard constraint may lead to an undesirable increase in the computational cost of a path planner due to the additional runtime required to regenerate infeasible solutions. A soft constraint has a lower computational requirement because it can be optimized over time. Although the use of a soft constraint can help to maintain a good balance between a path planner's solution quality and computational cost, a vehicle may bump into obstacles if its path is soft-constrained to an inadequate distance with obstacles. This may occur in practice due to the vehicle's actuation limitations or external forces, such as ocean currents.

Therefore, a combined constraint approach was developed and applied for obstacle avoidance to ensure a collision-free path while maintaining a reasonable computational load. In this approach, two parameters were used to control the obstacle avoidance behaviour:

1. A buffer distance d_{buff}, which was applied with a soft constraint.
2. A safety distance d_{safe}, which was applied with a hard constraint.

During a mission, the buffer distance can allow a vehicle to bump into the buffered obstacle, but with a penalty applied. It was used to address the concerns for sensor uncertainties in FLS measurements, as well as the limitations in vehicular actuation and the effect of external forces on the vehicle. On the other hand, the safety distance must be maintained by the vehicle to avoid collisions. In this combined constraint approach, the soft

constraint was configured to be stricter than the hard constraint. Thus, it reduces the tendency of solutions to violate the hard constraint, subsequently lowering the computational load of the path planner.

To achieve obstacle avoidance, the algorithm computes the minimum distance between a path and the obstacle detected by the vehicle. Assuming a solution X_i in a 3D Euclidean space, the problem involves an obstacle h with a detection point $O_{c,h} = (O_{cx}, O_{cy}, O_{cz})$ and a path segment that links two adjacent waypoints $p_{i,j}$ and $p_{i,j+1}$. Firstly, the nearest point on the segment $p_{i,j}\ p_{i,j+1}$ to the point $O_{c,h}$ was determined. The nearest point $\rho(s)$ can be parameterized by Equation (25). Next, vector projection was used to obtain Equation (26).

$$\rho(s) = p_{i,j} + s\left(p_{i,j+1} - p_{i,j}\right) \tag{25}$$

$$\hat{s} = \frac{\left\langle O_{c,h} - p_{i,j}\ ,\ p_{i,j+1} - p_{i,j} \right\rangle}{\left\| p_{i,j+1} - p_{i,j} \right\|^2} \tag{26}$$

To find the parameter s that gives the minimum distance within the path segment, Equation (27) was used to clip s to the range of [0,1]. Thus, the minimum distance from $O_{c,h}$ to $\rho(s)$ can be determined by using Equation (28), which gives the obstacle distance D_{obs} between the path segment and the detected obstacle.

$$s' = \min\left(\max\left(\hat{s}, 0\right), 1\right) \tag{27}$$

$$D_{obs}(s) = \left\| \rho\left(s'\right) - O_{c,h} \right\| \tag{28}$$

A hard constraint was applied to D_{obs} by using the safety distance d_{safe}. A feasible path must maintain a distance greater than d_{safe} from all obstacles. A solution is deemed infeasible and will be regenerated if any of its path segments contains $D_{obs} < d_{safe}$. Additionally, the solution was soft-constrained by using the buffer distance d_{buff}. For each path segment with $D_{obs} < d_{buff}$ and $D_{obs} \geqslant d_{safe}$, the penalty function F_2 in Equation (29) can be applied to obtain the penalty cost, which is inversely proportional to D_{obs}. The total penalty of a solution can be given by Equation (30).

$$F_{2,h}(p_{i,j}) = \frac{d_{buff} - D_{obs}}{d_{buff}} \tag{29}$$

$$F_2(X_i) = \sum_{h=1}^{H} \sum_{j=1}^{m-1} F_{2,h}(p_{i,j}) \tag{30}$$

In dynamic and unknown environments, a detected obstacle does not necessarily remain stationary in the same location; it may move and become irrelevant over time. Moreover, the memory of the path planner for storing obstacle detections may become overloaded with time, leading to an increasing latency in the planner as the mission progresses. Hence, the proposed algorithm must maintain a list of relevant detection points by removing irrelevant detections based on their elapsed time and distances to the vehicle. Detections that exceeded the distance $D_{obs,max}$ or the elapsed time $t_{obs,max}$ were removed.

$$D_{obs,max} = 2 \cdot D_{fls} \tag{31}$$

$$t_{obs,max} = \frac{D_{obs,max}}{V_a} \tag{32}$$

where D_{fls} is the detection range of the vehicle's FLS sensor. The following pseudocode describes the obstacle avoidance approach of the path planner. The process can be illustrated in Figure 3.

For each obstacle detection point $O_{c,h}$, where h = 1, 2, …, H,
  For each discretised waypoint $p_{i,j}$, where j = 1, …, m-1,
    Find s using Equation (26).
    Clip s to [0,1] using Equation (27).
    Find D_{obs} using Equation (28).
    If $D_{obs} < d_{safe}$
      Break and regenerate solution.
    Else if $d_{safe} \leqslant D_{obs} < d_{buff}$
      Apply penalty using Equation (29).
    Else
      $F_{2,h}(p_{i,j}) = 0$
    End if
  End for
End for

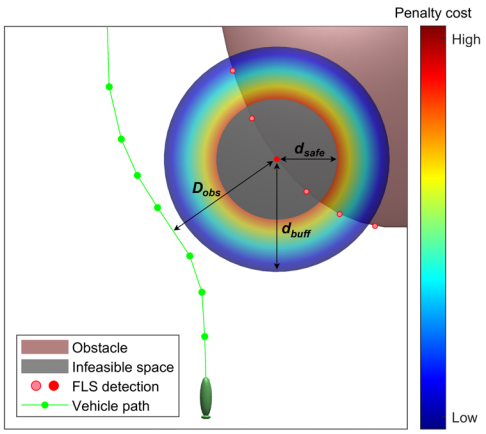


Figure 3: Example of obstacle avoidance scenario using one of the FLS detection points.

5 Vehicle and Sensor Models

In this study, the UTAS Explorer AUV "*nupiri muka*" was used as the test platform for verifying the proposed path planning system. The actuators of the AUV include a two-bladed propeller, a set of x-form rudders, and a pair of hydroplanes. The AUV is equipped with an FLS sensor, two ADCP sensors, and a dual-frequency side-scan sonar. The modularity of the vehicle allows additional sensors to be configured as required by its missions. The control architecture of the vehicle is based on the MOOS-IvP framework. The vehicle is also equipped with a backseat driver system, which enables seamless information/data exchange between the vehicle and its operators by using the MOOS-IvP middleware. HIL tests of the vehicle were facilitated by the backseat driver.

To conduct HIL tests on dry land, the AUV's motion response was simulated using an empirical model of the "*nupiri muka*" in the uSimExplorer application. The vehicle model can propagate the AUV's position and orientation based on the deflection angles of control surfaces and the propeller rotation rate, which were provided by the real-time actuator feedback from the AUV's hardware during the test. To simulate the effect of ocean currents, the vehicle model considered the drift force caused by time-varying currents by subscribing to the drift vector published by the uSimCurrent application. The vehicle model was governed by Equations (33) – (42).

$$v_t = \min\left(v_{\max} \cdot \frac{rpm_t}{rpm_{\max}},\ \dot{v}_{\max} \cdot dt \right) \tag{33}$$

$$\Delta\Psi_t = \min\left(\frac{-\alpha 3_t + \alpha 4_t - \alpha 5_t + \alpha 6_t}{4} \cdot \frac{\dot{\Psi}_{\max}}{\alpha_{\max}} \cdot \sqrt{\frac{rpm_t}{rpm_{\max}}},\ \dot{\Psi}_{\max} \right) \tag{34}$$

$$\Psi_t = \Psi_{t\text{-}1} + \Delta\Psi_t \tag{35}$$

$$\Delta\Theta_t = \min\left(\frac{-\alpha 3_t - \alpha 4_t - \alpha 5_t - \alpha 6_t}{4} \cdot \frac{\dot{\Theta}_{\max}}{\alpha_{\max}} \cdot \sqrt{\frac{rpm_t}{rpm_{\max}}},\ \dot{\Theta}_{\max} \right) \tag{36}$$

$$\Theta_{\dagger} = \Theta_{\dagger\text{-}1} + \Delta\Theta_{\dagger} \tag{37}$$

$$v_{hor} = \cos(\Theta_{\dagger}) \cdot v_{\dagger} \tag{38}$$

$$v_{ver} = -\sin(\Theta_{\dagger}) \cdot v_{\dagger} \tag{39}$$

$$x_{\dagger} = x_{\dagger\text{-}1} + \left[\sin\left(\Psi_{\dagger}\right) \cdot v_{hor} + drift\_x_{\dagger} \right] \cdot d\dagger \tag{40}$$

$$y_{\dagger} = y_{\dagger\text{-}1} + \left[\cos\left(\Psi_{\dagger}\right) \cdot v_{hor} + drift\_y_{\dagger} \right] \cdot d\dagger \tag{41}$$

$$depth_{\dagger} = \max\left[0,\ depth_{\dagger\text{-}1} + \left(v_{ver} - v_{buoy} \right) \cdot d\dagger \right] \tag{42}$$

where $\dagger$ is the current time and $v_{\dagger}$ is the vehicle speed, which can be resolved into a horizontal component v_{hor} and a vertical component v_{ver}. The vehicle position can be given by $x_{\dagger}$, $y_{\dagger}$, and $depth_{\dagger}$. The vehicle heading and pitch angle are denoted by $\Psi_{\dagger}$ and $\Theta_{\dagger}$, respectively. $rpm_{\dagger}$ is the propeller rotation rate, whereas $\alpha n_{\dagger}$ is the deflection angle of the n^{th} control surface. The drift velocity in the x and y direction are represented by $drift\_x_{\dagger}$ and $drift\_y_{\dagger}$. The kinematic parameters of the model are given in Table 2 and were based on the AUV's performance data, which were obtained from its field experiments [54, 55].

Table 2: Kinematic parameters of the vehicle model.

Parameter		**Value**
$v_{\max}$	Maximum body-fixed velocity	2.2 m/s[1]
$\dot{v}_{\max}$	Maximum rate of change of body-fixed velocity	0.1 m/s[2]
$\dot{\Psi}_{\max}$	Maximum rate of change of vehicle heading	5°/s
$\dot{\Theta}_{\max}$	Maximum rate of change of vehicle pitch	3°/s
$rpm_{\max}$	Maximum achievable propeller rotation rate	300 rev/min
$\alpha_{\max}$	Maximum deflection angle of control surfaces	22°
v_{bouy}	Static upward velocity due to buoyancy	0.06 m/s

The vehicle model was validated against the field data of the *nupiri muka* AUV through turning circle tests, which required the vehicle to enter a steady turn at a constant speed. The experimental data were collected in the field with the AUV conducting a full-rudder (23°) circle at a vehicle speed of 1.5m/s. The turning circle of the vehicle model was obtained from the HIL setup using the same rudder angle and vehicle speed. The two trajectories produced a similar turning radius with a discrepancy of less than 5%.

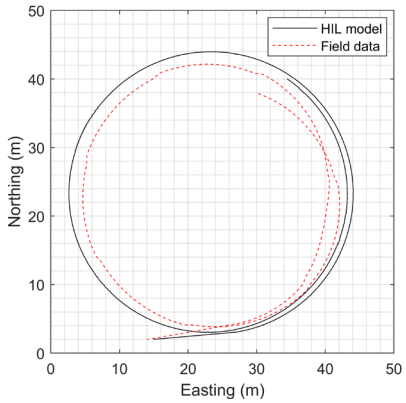


Figure 4: Turning circle test of the vehicle model.

The vehicle used an FLS sensor for the detection of obstacles. Based on the specification of the "nupiri muka", the FLS model were configured as follows: 80 metres detection range, 120° field of view, 121 beams (with 1° separation between beams), and 100Hz frequency. The configuration of an FLS sensor on an AUV depends on the requirements of its mission. For missions such as area-coverage surveys, it is more appropriate to use a horizontal sonar configuration, which aligns the sonar fan of FLS on the vehicle's horizontal plane. The vertical configuration of FLS is suitable for missions that require the AUV to operate near seabed or underneath ice shelves.

Hence, this study considered two sonar configurations as illustrated in Figure 5. For the vertical sonar configuration, an offset of 20° above the horizontal plane was used. During the HIL test, a priori unknown and irregular obstacles were present in the problem space. The FLS model can detect the boundaries of obstacles and generate the coordinates of the detection points. This information was recorded by the path planner for path computation.

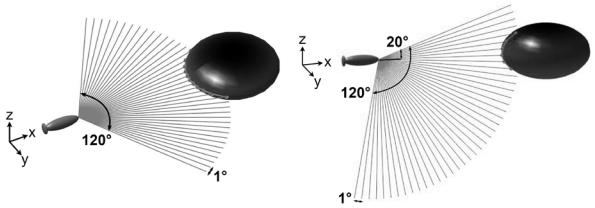


Figure 5: FLS sensors configured in horizontal (left) and vertical (right) planes.

The path planner required a current profile of the environment to generate time-optimal paths. A forward-looking H-ADCP sensor was simulated to provide the path planner with continuous information of current profiles. The H-ADCP model was a 300kHz sensor with 200 metres detection range. It can reconstruct a current profile of 200 metres × 50 metres in the vehicle's forward-looking region [56].

6 Experiments and Results

The implementation of the SDEQPSO path replanner using the MOOS-IvP framework was evaluated by performing a HIL test on the "*nupiri muka*" AUV. The payload computer was a Linux machine with Ubuntu 18.04 (GNU g++ 7.5.0) and Intel Core i5-6300U (2.4GHz CPU, 8GB RAM). The problem space was a virtual ocean environment with 1000×1000 square metres for 2D and 1000×1000×50 cubic metres for 3D. A priori unknown obstacles and time-varying ocean currents were present in the problem space. The virtual obstacles were placed in such a way that they will potentially block the AUV paths. Dynamic obstacles were configured to drift at random speeds up to 0.1 m/s in different directions. Field data of current profiles with added Gaussian noise were used to generate a time-varying current field with current velocity up to 0.5 m/s. The field data were sampled at Beauty Point, Tasmania, Australia from the ADCP sensors of the "*nupiri muka*" AUV.

Based on the properties of the "*nupiri muka*" AUV, the safety distance and buffer distance required for obstacle avoidance were defined as 10 metres and 30 metres, respectively. The population size of the SDEQPSO algorithm was configured as 150 particles. The algorithm parameters were configured based on the suggestions in Section 3. Different test cases as described in Table 3 were conducted during the HIL test. Monte Carlo methods with 20 runs were applied for every test case.

Table 3: Setups of HIL test cases.

Test Case	Dimension	FLS configuration	Obstacles	r_d (m)	φ_{max} (°)	$\theta_{\max}$ (°)
1	2D	Horizontal	Stationary	50	60	-
2	2D	Horizontal	Moving	50	60	-
3	3D	Horizontal	Stationary	50	60	5
4	3D	Horizontal	Moving	50	60	5
5	3D	Vertical	Stationary	50	5	20
6	3D	Vertical	Moving	50	5	20

The robustness of the SDEQPSO path replanner was evaluated using Monte Carlo methods under scenarios with stochastic processes, i.e., randomly moving obstacles and constantly changing ocean currents. To adapt an AUV path to ocean currents, the path replanner can accept current profiles in the form of real-time ADCP data or a predictive ocean model. A comparison was conducted between the path replanners that used different current data:

1. Real-time current profile inferred from the H-ADCP measurements,
2. Pre-generative current profile loaded directly from an ocean model.

This study also includes a comparison with two other planners to evaluate the performance of the path replanner:

1. An SDEQPSO-based path replanner without adaptation to ocean currents,
2. A local path planner that used the MOOS-IvP inbuilt dynamic obstacle manager [57].

The HIL test of the "*nupiri muka*" was conducted under the scenario described in Figure 6 and Figure 7. The AUV mission was to traverse the unknown, cluttered, and dynamic ocean environment towards a target while keeping a safe distance with the virtual obstacles and exploiting the favourable currents that can assist the vehicle motion. In the 2D tests (Cases 1 and 2), the target and the obstacles were placed on the ocean surface. For the 3D tests (Cases 3, 4, 5, and 6), the target was located at a depth of 15 metres, while the obstacles were placed at various depths between the surface and the target depth.

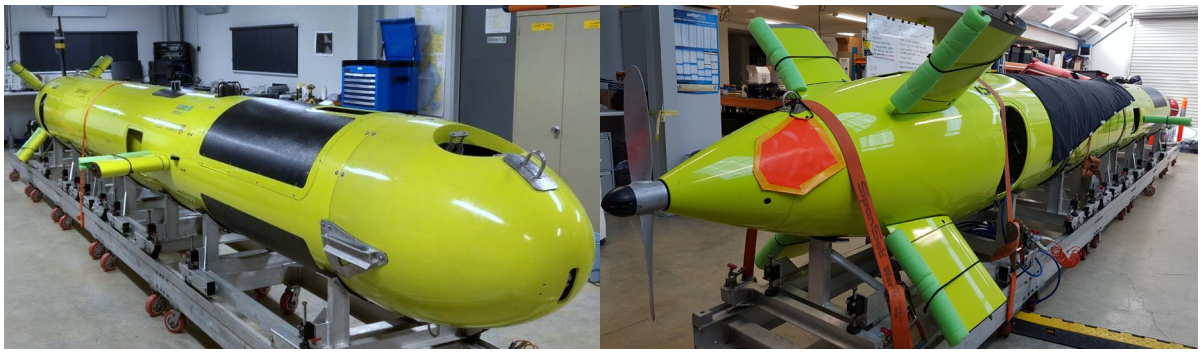


Figure 6: Bow view (left) and stern view (right) of the "*nupiri muka*" AUV.

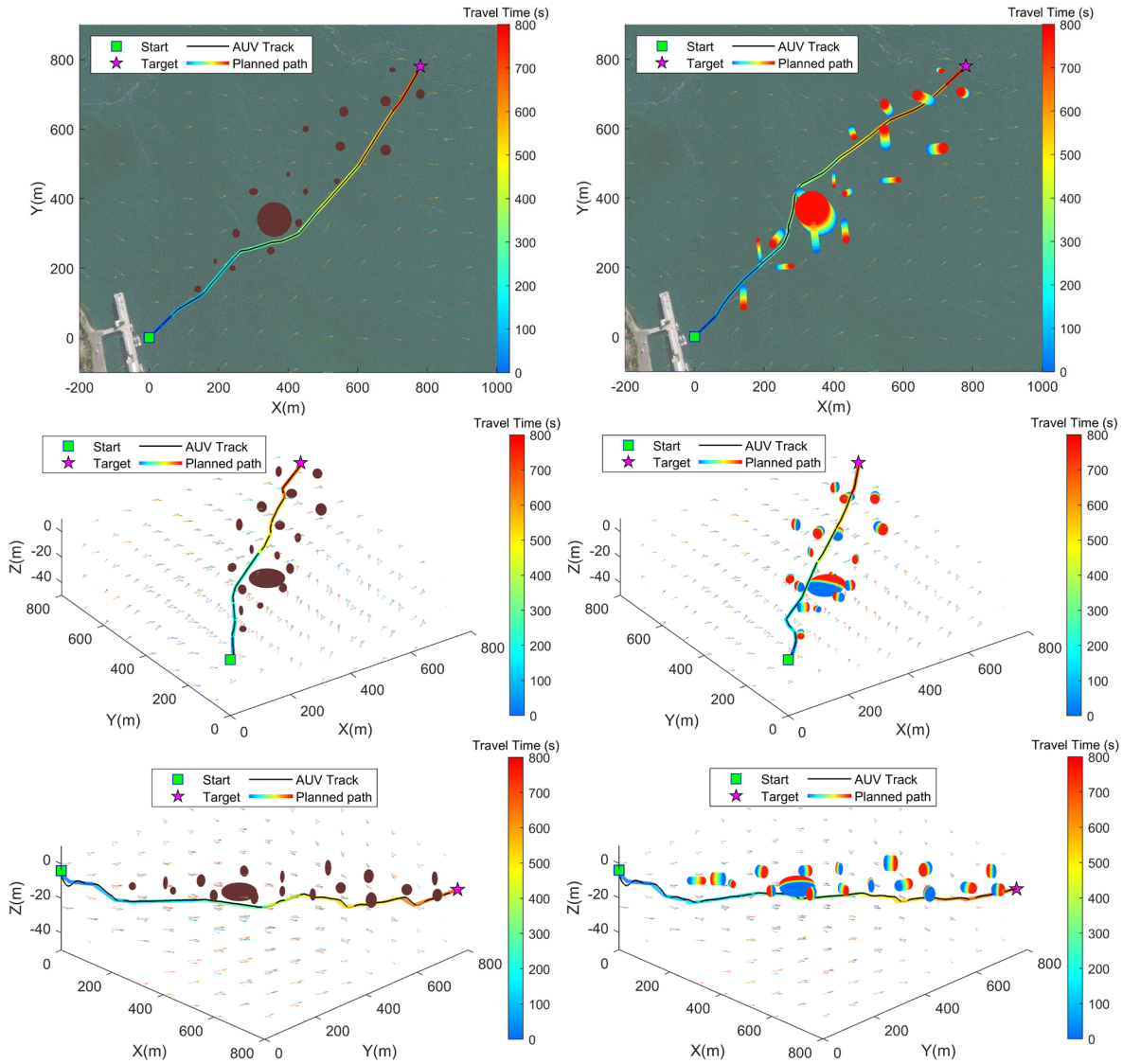


Figure 7: Path solutions of Case 1 (top-left), Case 2 (top-right), Case 3 (mid-left), Case 4 (mid-right), Case 5 (bottom-left), and Case 6 (bottom-right).

In Figure 7, the elapsed time during the tests is shown by the colour bars. The planned paths and vector fields of ocean currents are coloured according to the elapsed time. The static obstacles (Case 1, 3 and 5) are coloured brown, while the moving obstacles (Case 2, 4 and 6) are represented by the trails with colours corresponding to the elapsed time. A path is safe if it does not intersect with the brown static obstacles or the trails of moving obstacles with the same colour. The virtual obstacles were configured to obstruct the vehicle path intentionally so the AUV must use its sensors to detect and detour around the obstacles by replanning its path. When the vehicle used a horizontal sonar configuration (Case 3 and 4), the path replanner required it to manoeuvre around obstacles mainly by using yaw motion. Conversely, the planned path for the vehicle that used a vertical sonar configuration (Case 5 and 6) involved mostly pitch motion.

Throughout the experiment, the AUV was required to use its onboard controller and physically actuate its control surfaces and propeller in order to follow the planned path. In all test cases, the AUV maintained an average vehicle speed of 1.5 m/s. Figure 7 shows that the resultant paths are safe and collision-free. The executed vehicle tracks (black lines) showed close resemblances to the planned paths, indicating that the planned paths can be followed closely by the AUV in real time. By following a time-optimal path, the vehicle was driven to surf the favourable currents and to keep away from the adverse currents that would oppose its motion.

The path replanner must replan the AUV path in real time whenever its replanning flags were triggered, e.g., the previously planned paths conflicted with obstacles detected by the FLS sensor. Figure 8 shows the variation of runtime for the path replanner to replan the vehicle path during the real-time HIL tests. In the boxplot, the lower and upper ends of the boxes indicate the first and third quartiles of the runtime respectively, while the red lines across the boxes show the medians. During the HIL test, the runtime required for path replanning varied from milliseconds up to a maximum of 2.7 seconds (Case 3). The means of runtime in all test cases were found to be below 1 second. The resulting runtime shows that the path replanner can be run smoothly on the payload computer or an embedded system that uses the MOOS-IvP middleware.

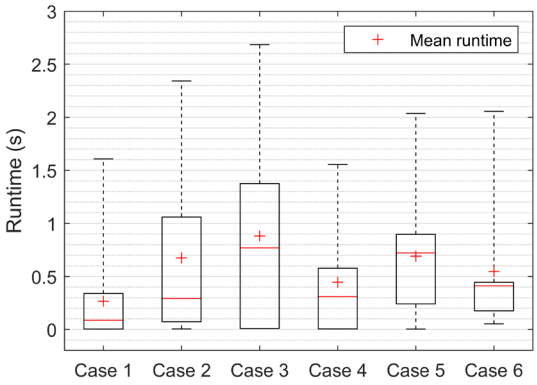


Figure 8: Runtime of the implemented SDEQPSO path replanner for replanning paths in different test cases.

The feasibility of the paths generated by the path replanner was checked against the physical limitations of the AUV actuators. The "*nupiri muka*" AUV has a minimum turning radius of 10 metres and a maximum pitch of 40° in the worst-case scenario. Figure 9 shows that the curvature radius of the generated paths was maintained higher than the minimum turning radius of the vehicle in the test cases. The variation of vehicle pitch in Figure 10 indicates that the required actuation was well within the vehicular limitation. Based on Figure 10, the vehicle mainly manoeuvred by using pitch motion when the vertical sonar configuration was used (Cases 5 and 6). Whereas, when the horizontal sonar configuration was used, the vehicle utilized minimum pitch motion (Cases 3 and 4).

To examine the path following performance of the AUV, Figure 11 shows the resultant cross-track error between the planned path and the actual path executed by the vehicle. Based on Figure 11, the variation of cross-track errors can be correlated with the occurrence of path replanning. In the test cases, increases in the cross-track errors were observed mainly when a path change was required after the AUV received a replanned path. The cross-track errors were found to be less than 0.2% of the total length of the path travelled by the vehicle. This corresponds to a maximum error of 2 metres, which is deemed acceptable given the total distance travelled and the size of the Explorer AUV (7.5 metres long). The safety distance (10 metres) and buffer distance (30 metres) used by the path replanner were able to tolerate the resultant cross-track error. The results showed that the paths generated by the path replanner can be safely followed by the vehicle under the influence of drift caused by ocean currents. Thus, the practicability of the SDEQPSO path replanner to generate feasible AUV paths in real time was verified.

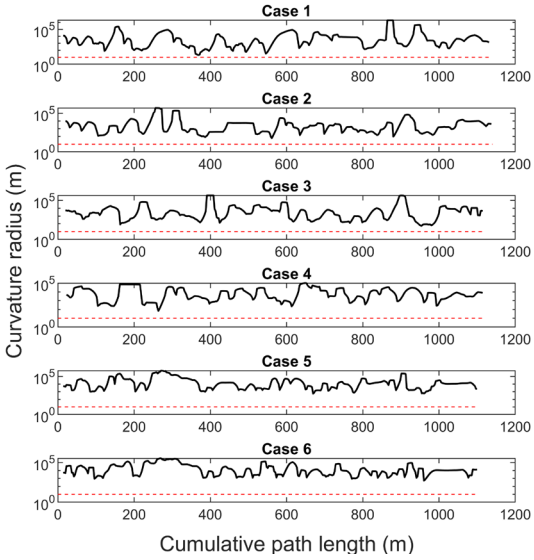


Figure 9: Variation of path curvature radius with respect to physical limitations.

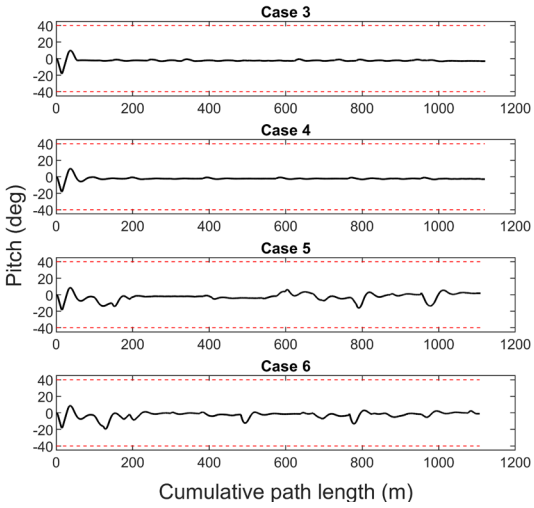


Figure 10: Variation of vehicle pitch with respect to physical limitations.

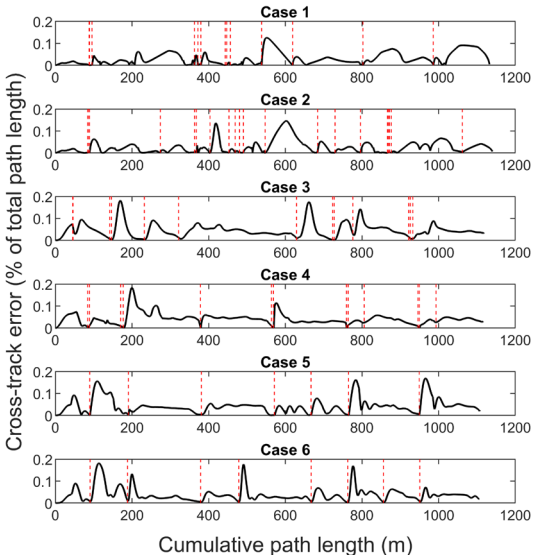


Figure 11: Variation of cross-track error between executed paths and planned paths (red dashed lines indicate when the vehicle received a replanned path).

To evaluate the performance of the proposed path replanner, the travel time required by the AUV to arrive at the target by following the paths generated by different path planners is collated in Figure 12. The local path planner was configured to follow a straight-line path that linked the starting point and the target. During the missions, the local path planner made local deviations to the planned path to avoid detected obstacles as required, and it returned the path after avoiding the obstacles. Cases 5 and 6 for the local path planner were excluded because the planner is only capable of making local deviations by changing the vehicle's heading, and thus, it is not compatible with the vertical sonar configuration.

The results in Figure 12 indicate that the proposed path replanner produced a shorter travel time than the local path planner. The path replanner reduced the AUV's travel time by 4 – 16% in 2D scenarios (Cases 1 and 2) and by 3 – 12% in 3D scenarios (Cases 3 and 4). The effectiveness of the path replanner in reducing the travel time varies depending on the type of current profile data used. The local path planner resulted in a longer travel time because it did not consider the optimality of the path; instead, it simply followed a straight-line path and deviated from the path to avoid obstacles. After passing the obstacles, instead of continuing to head towards the target, the local path planner always causes the vehicle to steer back to the original path regardless of its position. Furthermore, the solutions of the local path planner cannot be adapted to ocean currents.

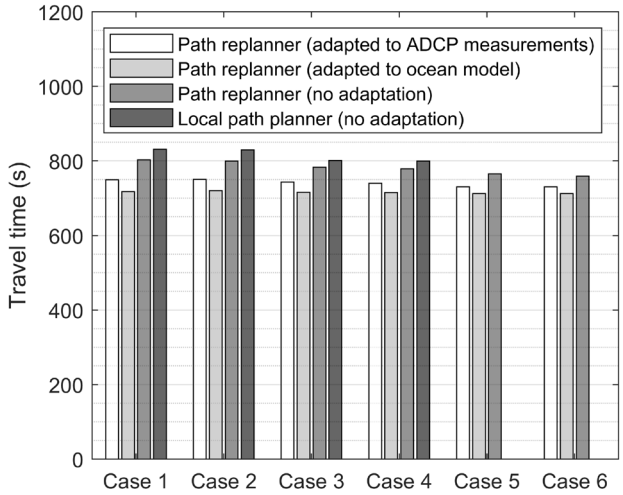


Figure 12: Travel time obtained by different path planners.

To examine the impact of ocean currents on the vehicle performance, the path replanner was configured to use different types of current profile data for adapting its solutions to currents. Contrary to the current-adapted path replanner that generated time-optimal paths, when the path replanner was configured without the adaptation to ocean currents, it simply searched for the path with the shortest distance towards the target. As shown in Figure 12, the resultant travel time of the two current-adapted path replanners was shorter in all test cases when compared to the path replanner that was not adapted to currents. By travelling on time-optimal paths, the AUV's travel time was reduced by up to 7% when the ADCP measurements were used, and up to 12% when the predictive ocean model was used. Travelling on the shortest-distance path, which did not consider the effect of currents, required a longer travel time to reach the target because the vehicle did not attempt to exploit favourable currents or to avoid adverse currents that opposed its motion and pushed it away from its path.

Between the two current-adapted path replanner, the predictive ocean model resulted in better solutions than the real-time ADCP measurements in all test cases. Figure 12 shows a maximum of 5% reduction in travel time was achieved by the path replanner that used an ocean model. Although the ADCP sensor measures ocean currents in real time, its profiling range is limited, thus leading to less current profile information available for path planning. Despite not having a real-time accuracy, a predictive ocean model can further improve the effectiveness of time-optimal paths because it provides a more comprehensive current profile for path planning. This enables the path replanner to further exploit ocean currents for every component of its time-optimal path. However, predictive ocean models are not always available, especially when planning an AUV path in unknown underwater environments. The proposed path replanner provides versatility in accepting both types of current profile data to generate a time-optimal path.

7 Conclusion

In this study, a path replanner was implemented as an independent application module in an AUV system that used the MOOS-IvP architecture. The implemented path replanner was verified in a HIL test of an Explorer AUV to analyse its interaction with the onboard controller and physical actuators. A variety of sensor configurations and test scenarios were involved in the experiment, which required the AUV to traverse across an unknown, dynamic and cluttered ocean environment. The proposed path replanner demonstrated its scalability for missions that require different sensor configurations and its versatility to accept different current profile data. During the experiment, the path replanner seamlessly worked in conjunction with the hardware on the test platform in real time to continuously generate a time-optimal path that exploited ocean currents and maintained obstacle avoidance. The experimental results verified that the planned path can be followed closely by the AUV under the disturbance of ocean currents. By comparison with a local path planner, the path replanner improved the vehicle performance by reducing up to 16% of its travel time required to reach a target. When compared to the shortest-distance paths, the generated time-optimal path reduced the travel time by up to 7% when onboard sensor measurements were used, and up to 12% when a predictive ocean model was used. By reducing an AUV's travel time and energy usage for transiting between survey locations, the path replanner can enable the vehicle to preserve its energy for conducting surveys. This reduces the necessity of retrieving and redeploying an AUV for recharging its battery during a long-duration mission, hence improving the vehicle's mission efficiency.

The next step for verifying the proposed path planner is to conduct in-water experiments. The path planner was implemented in MOOS-IvP to promote the ease of transition into field operations. The path planner can be applied to improve the competence of marine vehicle systems other than an AUV if the vehicle is equipped with a current profiler or when a predictive ocean model is available. The modularity of the implemented path planner ensures its scalability and extendibility for different vehicle systems.

8 Acknowledgement

This research was supported by the Australian Research Council's Special Research Initiative for Antarctic Gateway Partnership (Project ID SR140300001). The first author was supported by a Tasmania Graduate Research Scholarship provided by the Australian Maritime College (AMC), University of Tasmania. The authors acknowledge Autonomous Maritime Systems Laboratory (AMSL) in AMC for the help in setting up the HIL experiments.

9 References

[1] Edwards JR, Smith J, Girard A, Wickman D, Lermusiaux PFJ, Subramani DN, et al. Data-driven learning and modeling of AUV operational characteristics for optimal path planning. In: OCEANS 2017 MTS/IEEE; Aberdeen, UK: IEEE; 2017. p. 1-5.

[2] Lolla T, Ueckermann M, Yiğit K, Haley PJ, Lermusiaux PF. Path planning in time dependent flow fields using level set methods. In: IEEE International Conference on Robotics and Automation (ICRA); Saint Paul, MN: IEEE; 2012. p. 166–73.

[3] Yao X, Wang F, Wang J, Wang X. Bilevel optimization-based time-optimal path planning for AUVs. Sensors. 2018;18:4167.

[4] McMahon J, Plaku E. Mission and Motion Planning for Autonomous Underwater Vehicles Operating in Spatially and Temporally Complex Environments. IEEE Journal of Oceanic Engineering. 2016;41:893-912.

[5] Yao X, Wang X, Wang F, Zhang L. Path following based on waypoints and real-time obstacle avoidance control of an autonomous underwater vehicle. Sensors. 2020;20:795.

[6] Larson J, Bruch M, Ebken J. Autonomous navigation and obstacle avoidance for unmanned surface vehicles. In: Unmanned systems technology VIII; Orlando, FL: International Society for Optics and Photonics; 2006. p. 7.

[7] Casalino G, Turetta A, Simetti E. A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields. In: OCEANS 2009 - Europe; Bremen, Germany; 2009. p. 1-8.

[8] Sun B, Zhu D, Yang SX. An optimized fuzzy control algorithm for three-dimensional AUV path planning. International Journal of Fuzzy Systems. 2018;20:597–610.

[9] Benjamin MR, Defilippo M, Robinette P, Novitzky M. Obstacle avoidance using multiobjective optimization and a dynamic obstacle manager. IEEE Journal of Oceanic Engineering. 2019;44:331-42.

[10] Candeloro M, Lekkas AM, Sørensen AJ. A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. Control Engineering Practice. 2017;61:41-54.

[11] Petres C, Romero-Ramirez M-A, Plumet F. Reactive path planning for autonomous sailboat. In: 15th International Conference on Advanced Robotics (ICAR); Tallinn, Estonia; 2011. p. 112-7.

[12] Haddadin S, Urbanek H, Parusel S, Burschka D, Roßmann J, Albu-Schäffer A, et al. Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. In: IEEE/RSJ International Conference on Intelligent Robots and Systems; Taipei, Taiwan; 2010. p. 3109-16.

[13] Duchoň F, Babinec A, Kajan M, Beňo P, Florek M, Fico T, et al. Path Planning with Modified a Star Algorithm for a Mobile Robot. Procedia Engineering. 2014;96:59-69.

[14] Naeem W, Irwin GW, Yang A. COLREGs-based collision avoidance strategies for unmanned surface vehicles. Mechatronics. 2012;22:669-78.

[15] Singh Y, Sharma S, Sutton R, Hatton D, Khan A. A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. Ocean Engineering. 2018;169:187-201.

[16] Carsten J, Ferguson D, Stentz A. 3D Field D: Improved path planning and replanning in three dimensions. In: IEEE/RSJ International Conference on Intelligent Robots and Systems; Beijing, China: IEEE; 2006. p. 3381-6.

[17] Redding J, Amin J, Boskovic J, Kang Y, Hedrick K, Howlett J, et al. A Real-Time Obstacle Detection and Reactive Path Planning System for Autonomous Small-Scale Helicopters. In: AIAA Guidance, Navigation and Control Conference and Exhibit; Hilton Head, SC; 2007. p. 6413.

[18] Vasile CI, Belta C. Reactive sampling-based temporal logic path planning. In: IEEE International Conference on Robotics and Automation (ICRA); Hong Kong; 2014. p. 4310-5.

[19] Belkhouche F. Reactive path planning in a dynamic environment. IEEE Transactions on Robotics. 2009;25:902-11.

[20] Belkhouche F, Bendjilali B. Reactive path planning for 3-D autonomous vehicles. IEEE Transactions on Control Systems Technology. 2012;20:249-56.

[21] Duguleana M, Mogan G. Neural networks based reinforcement learning for mobile robots obstacle avoidance. Expert Systems with Applications. 2016;62:104-15.

[22] Cui R, Yang C, Li Y, Sharma S. Adaptive Neural Network Control of AUVs With Control Input Nonlinearities Using Reinforcement Learning. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2017;47:1019-29.

[23] Lin C, Wang H, Yuan J, Yu D, Li C. An improved recurrent neural network for unmanned underwater vehicle online obstacle avoidance. Ocean Engineering. 2019;189:106327.

[24] Cheng Y, Zhang W. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing. 2018;272:63-73.

[25] Likhachev M, Gordon G, Thrun S. ARA*: Anytime A* with provable bounds on sub-optimality. In: Advances in Neural Information Processing Systems; Vancouver, Canada; 2004.

[26] Ferguson D, Stentz A. Using interpolation to improve path planning: The Field D* algorithm. Journal of Field Robotics. 2006;23:79-101.

[27] Ferguson D, Stentz A. Anytime RRTs. In: IEEE/RSJ International Conference on Intelligent Robots and Systems; Beijing, China; 2006. p. 5369-75.

[28] Park C, Pan J, Manocha D. Real-time optimization-based planning in dynamic environments using GPUs. In: IEEE International Conference on Robotics and Automation; Karlsruhe, Germany; 2013. p. 4090-7.

[29] Galceran E, Campos R, Palomeras N, Ribas D, Carreras M, Ridao P. Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. Journal of Field Robotics. 2015;32:952–83.

[30] Sun B, Zhu D. Three dimensional D* lite path planning for autonomous underwater vehicle under partly unknown environment. In: 12th World Congress on Intelligent Control and Automation (WCICA); Guilin, China: IEEE; 2016. p. 3248–52.

[31] Ma T, Li Y, Jiang Y, Wang R, Cong Z, Gong Y. A dynamic path planning method for terrain-aided navigation of autonomous underwater vehicles. Measurement Science and Technology. 2018;29:095105.

[32] Hernández JD, Vidal E, Moll M, Palomeras N, Carreras M, Kavraki LE. Online motion planning for unexplored underwater environments using autonomous underwater vehicles. Journal of Field Robotics. 2019;36:370–96.

[33] Bruce J, Veloso M. Real-time randomized path planning for robot navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems; Lausanne, Switzerland; 2002. p. 2383-8.

[34] Brock O, Khatib O. Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths. In: IEEE International Conference on Robotics and Automation; San Francisco, CA; 2000. p. 550-5.

[35] Hernández E, Carreras M, Antich J, Ridao P, Ortiz A. A topologically guided path planner for an AUV using homotopy classes. In: IEEE International Conference on Robotics and Automation; Shanghai, China; 2011. p. 2337-43.

[36] MahmoudZadeh S, Yazdani AM, Sammut K, Powers DM. Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms. Applied Soft Computing. 2018;70:929–45.

[37] Zhou H, Zeng Z, Lian L. Adaptive re-planning of AUVs for environmental sampling missions: a fuzzy decision support system based on multi-objective particle swarm optimization. International Journal of Fuzzy Systems. 2018;20:650-71.

[38] Biswas S, Anavatti SG, Garratt MA, Pratama M. Simultaneous replanning with vectorized particle swarm optimization algorithm. In: 14th International Conference on Control, Automation, Robotics and Vision (ICARCV); Phuket, Thailand; 2016. p. 1-6.

[39] Lv M, Yang C, Zhang S. Real-time Route Re-planning based on Modified Particle Swarm Optimization Algorithm. In: IEEE 2nd International Conference on Electronic Information and Communication Technology (ICEICT); Harbin, China; 2019. p. 143-6.

[40] Zeng Z, Sammut K, Lammas A, He F, Tang Y. Efficient path re-planning for AUVs operating in spatiotemporal currents. Journal of Intelligent & Robotic Systems. 2015;79:135–53.

[41] Zeng Z, Sammut K, Lian L, He F, Lammas A, Tang Y. A comparison of optimization techniques for AUV path planning in environments with ocean currents. Robotics and Autonomous Systems. 2016;82:61–72.

[42] Lim HS, Fan S, Chin CKH, Chai S. Performance evaluation of particle swarm intelligence based optimization techniques in a novel AUV path planner. In: IEEE OES Autonomous Underwater Vehicle Symposium; Porto, Portugal: IEEE; 2018. p. 1–7.

[43] Panda M, Das B, Subudhi B, Pati BB. A comprehensive review of path planning algorithms for autonomous underwater vehicles. International Journal of Automation and Computing. 2020:1-32.

[44] Lim HS, Fan S, Chin CKH, Chai S, Bose N. Particle swarm optimization algorithms with selective differential evolution for AUV path planning. International Journal of Robotics and Automation (IJRA). 2020;9:94–112.

[45] Lim HS, Chin CKH, Chai S, Bose N. Online AUV path replanning using quantum-behaved particle swarm optimization with selective differential evolution. Computer Modeling in Engineering & Sciences. 2020;125:33-50.

[46] Newman PM. MOOS - Mission Orientated Operating Suite (OUEL Report). Department of Engineering Science, University of Oxford; 2008. Available.

[47] Benjamin MR, Schmidt H, Newman PM, Leonard JJ. Nested autonomy for unmanned marine vehicles with MOOS-IvP. Journal of Field Robotics. 2010;27:834-75.

[48] Paull L, Saeedi S, Seto M, Li H. Sensor-driven online coverage planning for autonomous underwater vehicles. IEEE/ASME Transactions on Mechatronics. 2013;18:1827-38.

[49] Hudson J, Seto ML. Underway path-planning for an unmanned surface vehicle performing cooperative navigation for UUVs at varying depths. In: IEEE/RSJ International Conference on Intelligent Robots and Systems; Chicago, IL: IEEE; 2014.

[50] Ferri G, Munafò A, LePage KD. An autonomous underwater vehicle data-driven control strategy for target tracking. IEEE Journal of Oceanic Engineering. 2018;43:323-43.

[51] Sun J, Lai CH, Wu XJ. Particle swarm optimisation classical and quantum perspectives. Boca Raton, FL: CRC Press; 2012.

[52] Piegl L, Tiller W. The NURBS book. Berlin: Springer Science & Business Media; 2012.

[53] Lim HS, Fan S, Chin CKH, Chai S, Bose N, Kim E. Constrained path planning of autonomous underwater vehicle using selectively-hybridized particle swarm optimization algorithms. IFAC-PapersOnLine. 2019;52:315–22.

[54] Pyper W. Yellow submarine prepares for first Antarctic mission. Australian Antarctic Magazine 2018;(35):12–3. Available from: https://search.informit.org/doi/10.3316/informit.216707160549452.

[55] Spain E, Gwyther D, King P. Submarine ventures under Sørsdal Glacier. Australian Antarctic Magazine. 2019;(36):18–9. Available from: https://search.informit.org/doi/10.3316/informit.520443225029371.

[56] Garau B, Alvarez A, Oliver G. AUV navigation through turbulent ocean environments supported by onboard H-ADCP. In: Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA); Orlando, FL: IEEE; 2006. p. 3556–61.

[57] Benjamin MR. pObstacleMgr: Managing Vehicle Belief State of Obstacles. Massachusetts Institute of Technology; 2020. Available from: https://oceanai.mit.edu/ivpman/pdfs/app_pobstaclemgr.pdf.